Software and Systems Modeling (2026) 25:847–869
https://doi.org/10.1007/s10270-025-01349-3

SPECIAL SECTION PAPER

MDE for crop representations in smart farming digital twins: a reinforcement learning perspective

Pascal Archambault[1] · **Houari Sahraoui**[1] · **Eugene Syriani**[1]

Received: 15 January 2025 / Revised: 12 November 2025 / Accepted: 13 November 2025 / Published online: 8 January 2026
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2026

Abstract
Digital twins of complex systems often involve the collaboration of multiple stakeholders. These domain experts do not speak the same domain languages, and this problem worsens when no domain-specific representations are usable by the twin. In controlled environment agriculture, digital twins must characterize crops via multiple concerns. That is, the digital twin must convey actionable results to its operators, who must assess the phenological and morphological development of the crop. Crop models are typically built to represent either phenology or morphology, making it difficult to simulate consistent representations across concerns. To address this, we introduce a model transformation based on a combinatorial optimization problem and study the applicability of genetic and reinforcement learning algorithms. We validate our methodology on the refinement of simulated strawberry phenological states and compare the applicability, performance, and diversity of our algorithms.

Keywords Digital twins · Domain-specific modeling language · Model transformation · Reinforcement learning · Model-based optimization

1 Introduction

Advances in model-driven engineering (MDE) and digital twins have enabled the automation of complex systems with applications to manufacturing [67], avionics [72], and logistics [28]. Digital twins are virtual representations of real systems that evolve alongside their twin through a continuous feed of sensor and actuator data. Enabled by simulation and predictive models, they provide users with services that allow for visualization, what-if analysis, and automation.

Digital twins have the potential to make a significant impact in the agricultural domain when applied to smart farming [1]. The integration of virtual models that continuously evolve with the farm enables farmers to develop farming systems that are less vulnerable to outdoor growing conditions. For instance, controlled environment agriculture (CEA) aims to control every factor that influences crop development. This differs from traditional farming, where the crop is susceptible to the uncertain outside environment. The practice of CEA has led to the development of vertical farms [45], where crops are grown vertically to maximize the usage of space, forming a cyber-biophysical system (CBPS) [17]. Such a system is typically operated by a team of domain experts in energy management and agronomy, such as a thermodynamics engineer, an agronomist, and a grower. The thermodynamics engineer works with the agronomist to plan the energy load of the system, while the agronomist and grower work together to achieve production goals.

A key difficulty faced by the operators of the CBPS is the lack of explicit knowledge about the behavior of the system. Traditionally, agronomical literature studies the influences of growth factors such as temperature and photosynthetically active radiation in silos, whereas CEA requires understanding the relationship between most if not all growth factors to optimize production. Agronomical studies relevant to CEA are often unavailable due to their prohibitive costs, variance in cultivars, or intellectual property rights. As crops are biological entities with inherent spatial and temporal variability,

Communicated by Judith Michael, Andreas Wortmann, and Loek Cleophas.

✉ Pascal Archambault
pascal.archambault@umontreal.ca

Houari Sahraoui
sahraoui@iro.umontreal.ca

Eugene Syriani
syriani@iro.umontreal.ca

[1] DIRO, Université de Montréal, Montreal, QC, Canada

their behavior is constrained to their growth environment, and studies often fail to generalize. Further, the collection of crop data within the CBPS is very costly and time-consuming and relies on operators exploring potentially unsafe configurations of the system. Digital twins are of great value to smart farming, as operators can leverage the virtual models and predictive services provided by the digital twin to further study the relationship between the crop and its environment [5]. To be of use to its stakeholders, the digital twin must present actionable information about the behavior of the real system using domain-specific information. For instance, an agronomist optimizes the climate regimen to achieve specific production goals over months, whereas a grower optimizes the health of the crop via daily care. Both users require different levels of abstraction: the agronomist needs information about the crop phenology (i.e., the unobservable growth dynamics), while the grower needs information about the crop morphology (i.e., topology and structure).

To enable the consistent representations of different crop concerns within the CEA digital twin, we developed a model transformation chain that provides representations of phenology and morphology in a cultivar-agnostic way. The transformation chain, through calibrated heuristics, can be easily adapted to different crop cultivars and represents a wide variety of growth settings. By bridging the representation gap between models of phenology and morphology, the digital twin can provide advanced visualization services to enable collaboration between agronomists and growers, aiding in the derivation of optimal system configurations [13]. We previously showcased the applicability of our methodology to generate crop morphologies using genetic algorithms [6]. However, while genetic algorithms are able to identify nearly optimal solutions in our solution space, their performance is highly dependent on the selection of their initial population, their operators, and their calibration. Genetic algorithms do not reuse simulation data and must be re-executed for different instances of the optimization problem. As such, simulation model evaluation can be really costly, and it is of high interest to reuse real data incoming from the physical twin.

To this end, reinforcement learning algorithms are theoretically attractive due to their ability to learn complex dynamics alongside an evolving system. More specifically, off-policy reinforcement learning algorithms such as Q-Learning [41] and TD3 [30] include the concept of a replay buffer to store previous simulation evaluations for future reuse. Further, reinforcement learning agents can leverage techniques such as knowledge distillation [32] and transfer learning [62] to adapt to new environmental conditions. Most recent applications of reinforcement learning have been in the game-playing [8] and stochastic control [78] domains, but their structure as Markov decision processes enables novel ways to solve combinatorial multi-objective optimization problems [23]. To foster simulation data reuse in our model transformation, we propose to replace the genetic algorithm proposed in [6] with a reinforcement learning agent that controls the optimization process.

This article builds upon our prior publication [6]. We have added relevant reinforcement learning notions to the background, have added Sect. 6 to expand upon our reinforcement learning setting, have conducted new experiments with the TD3 learning algorithm in Sect. 7.3.2, and compare our new results to NSGA-III in Sect. 7.

The main contributions of this paper are:

(i) a combinatorial optimization-based model transformation that bridges the knowledge gap between growers and agronomists;
(ii) a simulation architecture that enables the use of reinforcement learning algorithms with the L-system formalism;
(iii) a comparison of TD3-based off-policy reinforcement learning algorithms with the NSGA-III genetic algorithm to solve a combinatorial optimization-based model transformation.

We validate our approach by reproducing an experiment from our prior publication [6], where we refine the abstract morphological state of vertically grown strawberries. Our results show that, while theoretically grounded, the application of off-line reinforcement learning algorithms remains limited in a combinatorial optimization context.

In Sect. 2, we provide the background information and in Sect. 3 introduce a running example motivating our model transformation. In Sect. 4, we outline our simulation methodology and explain how we derive the abstract morphological state. In Sect. 5, we detail the process of refining the abstract state into a concrete morphological state and describe how to generate its visualization. In Sect. 6 we go over the modeling considerations that motivate the use of certain reinforcement learning algorithms and introduce our model architecture. In Sect. 7 we validate our approach with an experiment on strawberries growing in a vertical farm, and discuss our results. We then review related works in Sect. 8 and conclude in Sect. 9.

2 Background

In this section, we first define key concepts of digital twins in smart farming and provide our domain-specific definition. We then provide definitions of crop phenology, morphology, and L-Systems to situate the reader within our agronomical context. The section ends with an overview of reinforcement learning and how it relates to the optimization problem of our model transformation.

2.1 Digital twins in smart farming

The concept of digital twin can be accredited to the work of Michael Grieve with John Vickers [33]. Digital twins have classically been understood to be a virtual representation of a real system that evolves alongside the system through a continuous data feed. Using this data, the models comprising the digital twin are updated to mirror the behavior of the real system, and through simulation and predictive services, the digital twin is able to analyze future system behavior. However, the original concept of digital twin has evolved beyond its manufacturing roots to include multiple domains of science and engineering [42]. This evolution has led to digital twin terminology to be increasingly domain-specific [11], and what constitutes a digital twin for one domain may not be understood as such in another. Thus, we situate our digital twin terminology in a smart farming context.

A recent conceptual framework [75] provides a typology of digital twins in smart farming. The different types of digital twins mostly differ in their actuation schemes, levels of autonomy, and purpose. Authors first introduce the *imaginary* digital twin, which models the requirements of non-physical objects. Then, physical objects are introduced to the system to provide *monitoring*, *predictive*, *prescriptive*, and *autonomous* digital twin classifications. Monitoring digital twins are usually called digital shadows [7] in other domains and provide a real-time digital representation of the real system that can be used to diagnose its behavior. Predictive digital twins integrate forecasting models to predict future states of the system, while prescriptive digital twins couple their forecasting ability with optimization algorithms to provide actuation recommendations, as shown in Fig. 1. Direct and closed-loop control over the real system is performed by autonomous digital twins. The distinction between prescriptive and autonomous is made to delineate the current applications of digital twins in agriculture. Digital twins are still in their infancy, and the technology has not reached the level of safety and trust that is required to remove the domain expert from the system operation.

Digital twins are of high interest to the domain of agriculture. By leveraging the remote-sensing and predictive capabilities of digital twins, farmers can monitor and control their operations remotely and analyze multiple farm operation plans to optimize their operations. Smart farming practitioners envision a holistic view of digital twins, whereas every object on the farm, including machinery, livestock, and crops, should be virtualized using remote sensing. Such digital twins comprise models of biological systems. Thus, their design must take into account modeling modalities inherent to the domain [17]. Aside from the limitations related to control trust and safety, biological entities are notoriously difficult to observe. For instance, agricultural measurements can be categorized as either destructive or non-destructive,

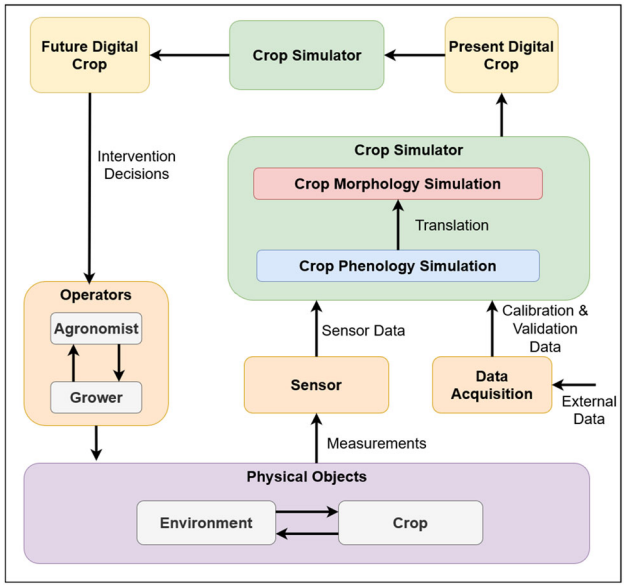

Fig. 1 Prescriptive digital twin conceptual framework, adapted from [75]

meaning that the crop must be destroyed to collect information about its state. Collecting destructive data is often a manual and time-consuming endeavor and is often required to calibrate and develop growth models. Further, collecting destructive data limits the ability of modelers to study the longitudinal behavior of the crop. On the other hand, non-destructive measurements such as the length and size of crop organs allow farmers to study the crop throughout its entire life but provide limited information on the dynamics of crop growth. By leveraging the real-time data flow stemming from the physical system, the digital twin can infer model parameters that would require crop destruction by using models based on non-destructive data.

2.2 Crop phenology

Crop phenology characterizes the underlying functions regulating crop development in regards to their environment. For example, crop photosynthetic activity and biomass allocation are phenological characteristics with dynamics that highly vary depending on crop genetics and growth environment. Agronomists often rely on specific crop phenological characteristics to categorize crop development into vegetative and reproductive stages. Measuring crop phenological phenomena involves laborious, expensive, and often destructive methods [14].

Models of crop phenology capture the underlying dynamics of crop growth in relation to its environment. Typically, such models comprise modules for the computation photosynthetic activity, soil nutrient exchange, evapotranspiration, and farming operations affecting the crop. These models are

used by agronomists to predict the impact of environmental recipes on phenological crop attributes such as the weight of organs, the distribution of chemicals within the crop, and water evapotranspired. Since the dynamics of the crop continuously evolve through time, phenology is described through systems of differential equations. Due to the complexity of their constituent parts, there are no analytical solutions that can model the entirety of crop phenology and its factors, and only a few general solutions exist for very specific characteristics [82]. Thus, simulation is often employed to approximate model results. For instance, the TOMGRO simulation model [43] involves 21 constants to characterize specific tomato cultivars. Typically, most model parameters must be calibrated through destructive measurements, killing the crop.

By describing the evolution of underlying crop functions across time, these models present actionable results to the agronomist via graphs and provide a basis for optimizing production plans. The agronomist can simulate different recipes and assess the simulated system traces to find optimal growth conditions. However, phenological models fail to provide actionable insights to the grower since their domain expertise lies on the visual assessment of physical crop characteristics, not on the understanding of underlying developmental phenomena. Thus, while models of phenology provide actionable results to the agronomist, there is a need to model the morphology to provide actionable results to the grower.

2.3 Crop morphology

Crop morphology defines the architectural and structural development of the crop, and includes functions such as branching patterns, and organ size and shape. Typically, most morphological characteristics do not necessitate destructive measurements, as it is possible to count the number of organs and measure their dimensions. It is the most appropriate level of abstraction for non-agronomists.

Models of crop morphology aim to capture the topology and relations between crop organs. They include cultivar-specific parameters, such as branch angle or reproduction thresholds to capture organogenesis and branching patterns. Also known as functional-structural plant models [77], they represent the physical characteristics of the crop and are useful for growers and agronomists to assess the evolution of the crop within their domain of expertise. Compared to crop phenology models that simulate underlying crop functions, morphological models simulate measurable and non-destructive crop attributes, which can be observed to characterize crop genotypes [10]. For instance, morphological models of tomatoes have been used to infer crop structural development from greenhouse images with L-Systems [53].

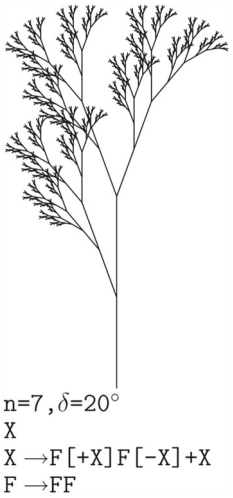

Fig. 2 L-System of a simple plant [65]. Parameter n specifies the number of derivation steps, and δ the branching angle at interpretation

2.4 L-System

The L-system formalism [51] models the growth processes of biological organisms, with an emphasis on crop development. It provides a framework for describing complex development patterns through the use of formal grammars and iterative, parallel rewriting rules and is a popular simulation formalism for crop morphology. Many variations of the formalism exist, but we will focus on parametric L-systems.

A parametric L-system is described as a tuple $G = \langle \Sigma, \omega, P, I \rangle$ where Σ contains the symbol of an alphabet, ω is the axiom, P is a set of parameterized production rules and I a set of interpretation rules. Starting from the axiom ω, the evolution of the system is simulated by repeatedly computing the new parameters and applying the production rules on the previous iteration string. Interpretation rules define how to interpret the derived L-System string according to the rules of turtle graphics [63]. Turtle graphics [2] is a method for visually interpreting and representing strings generated by L-systems. The method sets a cursor, or so-called turtle, on a blank canvas that interprets each symbol as specified by the interpretation rules to draw the appropriate shapes. The turtle reads the string iteratively, and when a separate context is found in the interpreted string, a new turtle starts drawing in parallel. This results in the efficient interpretation and rendering of L-System strings and links grammar-based formalisms to graphics. Figure 2 provides an example of a crop drawn by an L-System along with the model parameters and grammar.

2.5 MDE in smart farming

MDE enables the reliable design and implementation of decision support systems [58]. By providing multiple levels of abstraction, models allow system designers to capture more

or less complex phenomena to monitor and operate their systems. These multiple abstractions prove inherently valuable for the agronomical domain, as long as they are actionable by their stakeholders. To be relied upon, decision support systems for the optimization of crops should present results that correspond to the stakeholders' expectations [38].

Crop model suites, such as DSSAT [44], often present simulation results through graphs and charts that often fail to communicate the structural patterns informing the grower. Their parameterization involves making labor-intensive, costly, and destructive observations on the crop, which reduces the number of longitudinal data points, as they often model the dynamics at the crop level. On the other hand, L-System models are simple to calibrate through non-destructive measurements, but their output often do not capture key properties such as yield weight or other phenological characteristics. Typically, they are platform-specific and often implemented using different tools [77], making their integration with other models a challenge.

MDE principles can facilitate the development of decision support systems by integrating domain-specific languages and tools that leverage the knowledge of multiple stakeholders [17]. Rather than building large platform-specific models to integrate morphology and phenology, we can model the relationship between dynamics and structure to separate our concerns. That is, we can utilize existing mechanistic phenological models and couple them to a structural model characterizing genotypes to provide a holistic crop-level representation. As there is no known mechanistic relationship between crop phenology and morphology, we leverage model transformation chains to implicitly capture such interactions [27] via an optimization problem. While multiple optimization frameworks exist, this paper focuses on the application of reinforcement learning as an extension to our prior experiment using genetic algorithms [6].

2.6 Reinforcement learning

Reinforcement learning is an agent-based machine learning technique that seeks to maximize an arbitrary reward signal over a sequential decision-making process [70]. Given an initial state in the environment, the agent applies an action and observes the resultant changes along with its reward. Through repeated interaction with its environment, the agent explores and learns an optimal mapping, called *policy* π, between actions and states that maximize the reward signal. Formally, the learning process of an agent can be described by a Markov decision process $MDP = \langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$ where $s \in \mathcal{S}$ is the state space of the system, $a \in \mathcal{A}$ the set of possible actions, $P(\cdot|s, a) \in \mathcal{P}(\mathcal{S})$ the probability density function that $s \rightarrow s'$ given a, $\mathcal{R}$ the reward signal from the transition $s \rightarrow s'$ after taking action a, and $0 \leq \gamma < 1$ the discount factor.

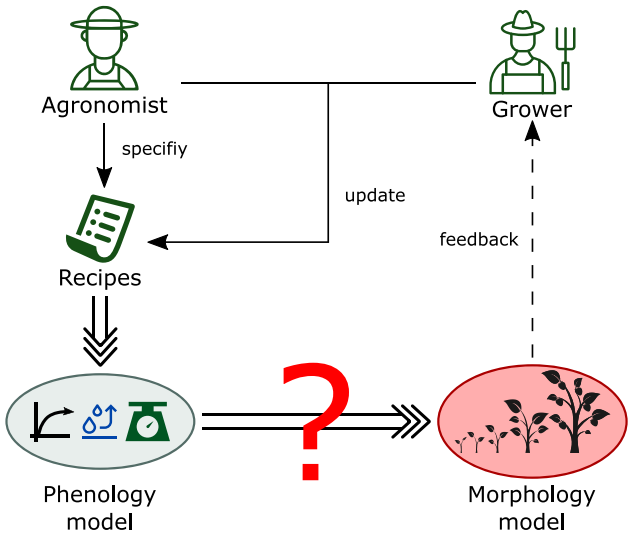

Fig. 3 Knowledge gap between agronomist and grower

At each iteration of the learning phase, the agent learns to explore the system by learning a surrogate of P via the policy π. The reward signal provided to the agent guides its search through learning, and gradient-based algorithms can be used to converge to a local optimum. Reinforcement learning approaches can better leverage exploration data from earlier algorithm iterations and have shown to be more efficient than genetic algorithms for certain design space exploration problems [46]. We instantiate and provide more context about the formulation of our specific MDP in Sect. 6.

3 Running example

In this section, we provide a running example to illustrate the complexity of operating a digital twin for the agronomist and grower. We detail the different domain-specific needs each user has for the digital twin, and highlight key difficulties in designing such a digital twin.

As a running example, let us consider the CEA of tomatoes with two stakeholders: an agronomist and a grower. CEA systems are complex to operate and involve the use of multiple cyber-physical systems to regulate ventilation, irrigation, radiation, chemical distribution, and pressure within the environment [66]. By maintaining direct control on the environment, agronomists and growers can adapt the growth conditions of crops to satisfy specific business needs. For instance, a tomato nursery aims to nurture and sell tomato seedlings, whereas a tomato farm wants to grow the most marketable crop yield. However, controlling the CEA system to achieve such business goals is no trivial task as the factors that influence the evolution of the system may be orthogonal. In the case of a nursery, crops are bred to develop their fruiting potential without inducing production while farms

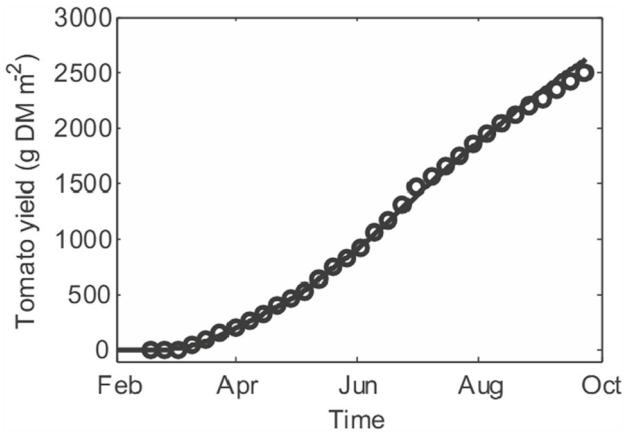

Fig. 4 Example phenological state produced by a tomato phenological model [73]. The circles represent real-world observations, and the line simulation results



Fig. 5 Concrete morphological state produced with a tomato L-System model [21]

seek to maximize yield, which requires conflicting environmental conditions. As there lacks domain understanding on the interactions between the crop and its controlled environment, the KPIs produced by the digital twin, such as energy consumption, or crop evapotranspired water and yield, can help operators gain deeper insight into the operation of their CBPS.

To operate the CEA system, the *agronomist* plans *environmental recipes* which define a production calendar detailing the environmental conditions necessary to achieve the desired yield outcome. The agronomist chooses the recipes based on their tacit knowledge, such as their understanding of crop developmental phases throughout a season, and the pheonological data available for specific tomato cultivars, such as the density of roots or chlorophyll content of leaves. Most of this data needs to be collected in a lab by destroying crops, which reduces potential production and removes longitudinal data points from the production cycle. Recipes typically do not exploit the full range of possible environmental conditions of the CEA system, as uncertainty brought by experimental recipes may lead to large discrepancies between the expected and produced yield. With the collected data, the agronomist calibrates and assesses the results from states produced by the phenological model. For instance, using weekly harvest data, the phenological model can be calibrated to estimate yield under specific conditions. The trajectory produced by the model is thus called the phenological state, as shown in Fig. 4.

While the agronomist handles the farm production plan and ensures that the farm reaches given production goals, the *grower* must ensure that the tomato crops receive the proper care throughout production. For example, the grower ensures that irrigation remains adequate, that the tomato crops are in good health, and prunes organs that negatively impact crop yield. Due to the inherent nature of crop development and the added system inertia caused by mass crop production, the grower must adjust the growing conditions daily, slightly deviating from the recipes provided by agronomists. The grower relies on their tacit knowledge, such as their understanding of branching patterns, the order of appearance of organs (i.e., organogenesis), and the general shape of the crop, to visually inspect and act upon it. Thus, the grower's job relies on observing non-destructive crop data, such as the number of tomato leaves, to perform actions like pruning. Just like humans do not form three arms, the structure of the tomato crop evolves through rules based on observable characteristics, which are captured by the L-System. Such models have reliably been used to simulate tomato crop morphology [48] as it embeds the well-formedness rules that characterize crop growth. An example morphological state for the tomato is shown on Fig. 5.

The operational complexity of CEA systems makes the interactions between the agronomist and grower crucial to achieving production goals. Both experts must work together to iterate on a production plan that meets business expectations and ensures healthy crop development. To achieve this, the domain experts rely on simulation-based decision support systems [4] to explore the state space of the CEA system and provide recommendations for its operations. Typically, such systems provide support for the agronomist via the simulation of phenological crop characteristics, such as photosynthesis and evapotranspiration, but fail to support the grower with a model of the tomato morphology. While tomato morphological models exist, their construction and usage is often decoupled from the underlying phenological phenomena that characterize crop growth [20]. By bridging the gap illustrated on Fig. 3 between the crop phenology and morphology, we aim to provide a consistent representation of the structure and dynamics of the crop that supports smart agriculture decision support systems.

A prescriptive digital twin of our CEA system would improve its efficiency in terms of yield (fruit production) and energy consumption (cost) by providing its operators recommendations that maximize business objectives. Based on a prior conceptual framework [75]. Figure 1 showcases the

components of our digital twin. We created a domain-specific language that enables mapping an instance of phenology to an instance of morphology by leveraging model transformations. Our model transformation is based on multi-objective optimization and finds instances of morphology that are closest to a derived optimal specimen. This article aims to investigate the viability of reinforcement learning algorithms, such as TD3, in solving our optimization problem and providing agronomists and growers with a method to assess the impact of the environment on the phenological and morphological characteristics of the crop.

4 Model transformation chain

We first present the overall architecture of our transformation chain, then describe how to produce a phenological state from a model of crop phenology. Finally, we propose an abstract morphological state and describe how to obtain it from a phenological state.

4.1 Overall architecture

The prescriptive digital twin of a smart farming system, as shown in Fig. 1, must represent the crop and simulate its possible future states to provide intervention decisions to operators. Models of phenology enable the digital twin to predict the impact of the environment on the phenological characteristics of the crop but fail to provide information on the crop structure. In contrast, morphological models fail to capture the impact of environmental conditions on the crop. Thus, we propose a methodology that bridges the gap between different aspects of crop growth.

To bridge the gap between crop representations, our architecture sequentially transforms models of different formalisms to achieve the graphical representation of a crop from a given set of growing conditions. The conditions become the input of a crop phenological simulator which outputs the phenological state of the crop. For instance, DSSAT [44] can be used to produce a phenological state, a time series of crop characteristics such as dry weight. This phenological state is then mapped to an abstract morphological state using a model transformation (i.e., translation). This abstract morphological state is represented using our DSL and specifies an ideal proportion of crop organs, which gets refined into a concrete morphological state through a multi-objective optimization-based model transformation. From the obtained optimal concrete state, we generate an animation of crop growth using the L-System model to convey the graphical results. Since the concrete morphological state is produced by an L-System model, the development of the crop should follow biological well-formedness rules that are not present in the abstract state. Following the methodology summarized on Fig. 6, we enable the generation of well-formed crop morphologies based on the simulation of crop phenology under specific conditions.

The phenological and morphological models capture specific, actionable KPIs for the agronomist and grower, which would include predicted yield and branching patterns in the case of a tomato crop. However, the different models typically do not work jointly to provide a consistent multi-view of the system. As such, the models may conflict in their results by projecting different organogenesis patterns or by providing mismatching estimates of the same KPI, such as leaf area index [12]. By coupling the morphological model to its phenological counterpart through an abstraction of crop morphology, we can present consistent results at multiple levels of abstraction to the grower and agronomist while producing specific KPIs that guide the farming system operation. In the case of a tomato greenhouse, this would include information about the evapotranspiration of the crop, its weight, and its height.

4.2 Producing the phenological state

Phenology is typically modeled as a dynamical system through differential equations, which describe the system through the evolution of continuous values, typically through time. However, we want to capture the state of development of the crop at discrete intervals to generate results at specific points in time. This can either be done through deriving an analytical solution to the system, which is unfeasible for most biological systems due to their complexity, or by using simulation to approximate the solution using numerical methods [31]. The simulator outputs a trace of model variables that are used to represent the phenological state of the crop at specific points in time.

4.2.1 Crop simulator

Simulation models of crop phenology simulate the continuous growth of the crop based on its environment. Such models incorporate modules that compute photosynthesis [83], water and chemical absorption [56], and the effects of temperature on crop development [50], and output crop growth phenological characteristics such as crop dry weight, evapotranspiration, and relative growth rate. Since crop simulation models vary in complexity and implementation, the mechanistic representations of phenological states vary. For the interested readers, we refer to a recent survey on crop yield models [19].

4.2.2 Phenological state representation

The phenological state encapsulates the results of the crop simulator, which can take many forms, such as an object or

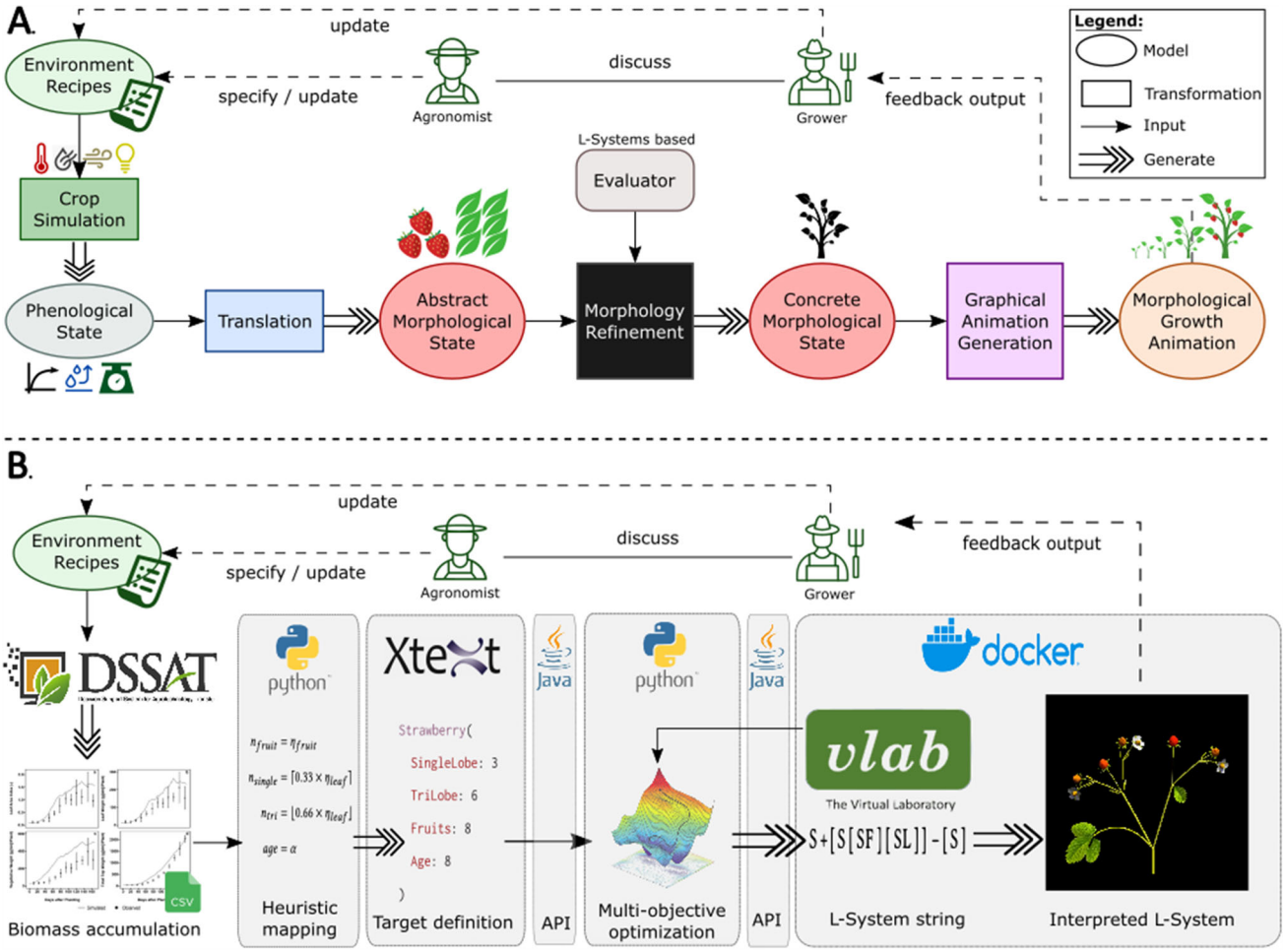

Fig. 6 Methodology for the generation of crop morphology from environmental recipes. Section A. presents our methodology and the multiple steps surrounding the transformation and section B. showcases the instantiation of our method to our problem

a time series of properties. We export the simulator output to a CSV file to integrate the simulation results with our optimization algorithm. This is done automatically by the crop simulator, but as simulation models vary in implementation, the file may need to be filled out manually by the agronomist. Each row of the file represents the phenology of the crop at every simulation time step, typically a day, and contains information about the crop. For the tomato example, the file might contain the age and weight of organs. The translation then uses the file to generate the abstract morphological state.

4.3 Generating the abstract morphological state

As crop simulation models typically aim to assess final crop yield, they often do not accurately capture the organogenesis patterns of every organ and their counts and are often unable to characterize the crop morphology. Thus, to bridge the gap between phenological and morphological characteristics, we generate an abstract morphological state from the simulated crop characteristics. To do this, we employ a user-defined translation heuristic that maps the phenological state to a concrete abstract morphological state based on the available phenological state representations output by the crop simulator. Certain phenological models directly compute the counts of organs of interest, which are directly translatable to an abstract morphological state. However, this is not the case for every phenological model, as relations between phenological attributes and observable morphological attributes must be defined, such as a mapping from organ weight to organ count.

4.3.1 Abstract morphological state

The abstract morphological state is an intermediary representation between the phenology and morphology of the crop that bridges the gap between models relying on destructive measurements such as crop weight and non-destructive measurements of crop organ counts. If we take the tomato example, the abstract morphological state describes the leaf count, tomato fruits, length of the main stem, and its age.

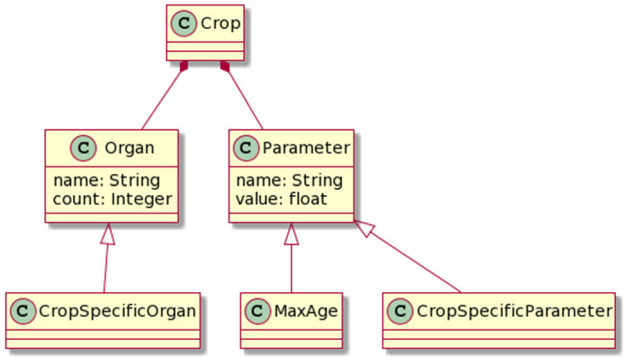

Fig. 7 Abstract Morphological State Metamodel

By introducing this abstract state between models relying on destructive and non-destructive measurements, we aim to characterize different crop genotypes by calibration of the translation heuristic.

The abstract state is implemented with a domain-specific language that captures the crop-specific parameters used for the morphology refinement. The language relies on user-defined crop modules that characterize the metamodel of the abstract morphological state, comprised of organ types and simulation parameters as shown in Fig. 7. The metamodel of the abstract morphological state relies on crop-specific organs and parameters that must be defined for individual crop species within a Java object loaded by the language. We provide such examples in the case studies. The abstract state is then refined through our optimization-based transformation to generate the appropriate concrete morphological state.

4.3.2 Translation

The translation heuristic maps the output of the crop simulator to the abstract morphological state and is derived by the agronomist based on their knowledge and experience with specific crop cultivars. As crop growth is highly sensitive to its environment, and as developmental characteristics depend on the crop cultivar and species, this heuristic function must be adjusted to each farming operation. To do this, agronomists must experiment and derive their parameters by collecting destructive crop measurements, i.e., measurements that kill the crop, such as vegetative dry weight, to correlate organ weight to organ count. If available, the agronomist can use a simulator to estimate destructive crop measurements. Once this mapping between organ weight and count is defined, we translate the phenological state to an abstract morphological state. The translation operation can be automated via an interface linking the outputs of the crop simulator with the abstract morphological state language. As agronomists are not computer experts, they can manually apply the translation using spreadsheet tools such as Microsoft Excel, from which they can manually input the abstract state using our domain-specific language.

5 Morphology refinement and growth representation

Given the large state space of the morphological model, which results from the numerous possible varying parameters, it is essential to explore this space efficiently to refine the abstract state into its concrete form. To achieve this, we model the space exploration as a multi-objective optimization problem and solve it using a genetic algorithm. Through this algorithm, we derive the parameters of the concrete morphological crop model, leading to the creation of a concrete morphological state, enabling the visualization of the crop morphology. Multi-objective genetic algorithms are particularly effective because they can identify Pareto-optimal solutions. From these solutions, we select the optimal one using a user-defined selection heuristic. This heuristic acts as a tie-breaker among the solutions, prioritizing specific crop organs.

5.1 Morphology refinement

To refine the abstract morphological state, we employ an optimization-based model transformation that uses a crop-specific L-System model to simulate possible morphological configurations of the crop and select which parameters lead to an optimal representation of the crop morphology. The optimal parameters of our morphological model are selected with a user-defined heuristic that prioritizes correct organ counts for specific organs.

5.1.1 Concrete morphological state

The concrete morphological state of the crop is the result of a parametric L-system model, which characterizes the structural and topological development of the crop by applying parallel string rewriting operations on a derivative string. This derivative string is then transformed to an interpretation string, as shown in Fig. 8, and interpreted by the turtle to visualize the crop. Our optimization-based transformation derives the parameters that generate a concrete morphological state closest to the specified abstract morphological state.

5.1.2 Crop specification

L-system models for different crops have different nomenclatures for organ names and types. To define the abstract morphological state of any crop, the DSL relies on crop-specific modules that define a mapping between the interpreted L-System string and the crop organs. The mapping

```
S+[S[SF][SL]]-[S]
```

Fig. 8 Example tomato L-System interpretation String. S denotes a stalk, a rotation, F a fruit, and L a leaf

pairs an organ with a regular expression used to extract crop organ counts from the string, which we use to optimize the L-system model parameters. For instance, the interpreted string of a tomato might contain the character `F` to represent fruits, which we count using a regular expression.

5.1.3 Optimization problem

To find the optimal concrete morphological state that characterizes the abstract state, we pose a multi-objective optimization problem where we minimize the difference between the generated organ count and the target specified by the DSL. For example, given a tomato crop with organs $o_i \in O$ such as o_{leaf} and o_{fruit} and their respective count n_i, and $C_i(S_j)$ a function evaluating the count of organ o_i for the candidate solution S_j for parameters $x_{veg}, x_{rep} \in X$, we define objective functions F_i counting the distance between the target organ count and candidate solution:

$$F_i(x_j) = |\lambda_i C_i(S_j) - n_i| \tag{1}$$

As crop morphological models may not simulate precise organ counts but rather the branching structure, we scale the count of organs i for a solution with parameter λ_i. This way, λ can be adjusted to minimize the impact of an over or underestimation of organ counts by the morphological model on the family of solutions.

We pose the following multi-objective optimization problem for each objective F_i:

$$\forall o_i \in O \min_{x_j \in X} F_i(x_j) \tag{2}$$

Our goal is to minimize the distance between the count of each organ geenrated by the l-system $o_i \in O$ and the target count specified by the abstract state. We explore the use of reinforcement learning to solve our multi-objective optimization problem in Sect. 6.

5.2 Crop visualization

When an optimal concrete morphological state has been found, we present it graphically in two possible ways: either by visualizing the final resulting crop and its evolution as a timeline of simulation snapshots, from which we generate an animation. The default interpretation of the L-System string generates the final state of the crop by interpreting the entire string using turtle graphics. Depending on the implementation of the morphological model, snapshots of the morphological states can be replayed to a specific threshold and displayed side-by-side to assess the evolution of the crop. The strings can be reinterpreted by the turtle step-by-step to generate an animation of the evolution of the crop. It is possible to view the entire animation at once or step through it.

5.3 Genetic algorithm

Our approach was first tested with metaheuristics [6], where we used the NSGA-III algorithm to solve our optimization problem [18]. This choice is motivated by the ability of NSGA-III to tackle multiple objectives with large amounts of parameters. We promptly illustrate the NSGA-III algorithm using the tomato example.

5.3.1 Algorithm

The tomato crop module provided to the DSL specifies which parameters need to be optimized, along with their constraints, if any. For example, we may wish to optimize parameters x_{veg} and x_{rep}, the vegetative and reproductive plastochron, to obtain a count for each organ $o_i \in \{fruit, leaf\}$.

The algorithm first encodes our parameters into random candidate vectors $\langle x_{veg}, x_{rep} \rangle \in X$. To bind the space of possible vector values, each parameter is discretized and encoded as a binary string. Then, the solution space is divided p times for each of our M objectives to create $H = \binom{M+p-1}{p}$ reference points on an M-dimensional normalized hyperplane. The algorithm then normalizes our solutions into the normalized hyperplane space. We refer the reader to [18] for details on this normalization and hyperplane construction. To maintain a diverse set of solutions, we set the number of divisions to $p = 15$.

Next, the algorithm selects k points closest to the H reference points. When a point is closest to multiple reference points, e.g., $k < H$, random candidates are randomly generated to be in the neighborhood of the most referenced point until a population P_1 is formed. From this population, an offspring population Q_1 is created by applying a two-point crossover operation and a mutation operator, each with a 50% chance. The mutation randomly increases or decreases the value of the encoded binary string with a 50% chance.

Once the population $P_1 \cup Q_1$ is obtained, N points are selected using a fitness function to form the next population P_2. To do this, the morphology of every candidate is simulated using the morphological model, and evaluated by summing each objective function F_i, which we minimize. To converge to an optimal solution, the algorithm iterates on the population h times, starting again from the construction of the hyperplane. The output of the genetic algorithm is a set of Pareto-optimal solutions S, containing optimal solution

vectors $\langle x_{veg}, x_{rep} \rangle_j$ to our optimization problem, which we reduce to a single optimal solution using a selection heuristic.

5.3.2 Selection heuristic

To prune our solution set, we apply a crop-specific heuristic function that prioritizes certain organs over others. Continuing with our tomato example, we may select a solution vector $\langle x_{veg}, x_{rep} \rangle$ that produces an optimal number of fruits over an optimal number of leaves. The agronomist typically chooses this heuristic function according to his knowledge of the crop cultivar. Formally, the function applies some preference over certain organ counts, that is, for organs $k_1, k_2 \in O$ where k_1 is preferred over k_2:

$$\min_{x^* \in X} F_{k_1}(x^*) \leq \min_{x^* \in X} F_{k_2}(x^*) \tag{3}$$

After applying our selection heuristic, we obtain the optimal set of morphological parameters from which we generate the visualization.

6 Application of reinforcement learning

In this section, we first instantiate our optimization problem to a Markovian process to satisfy the reinforcement learning setting. We then motivate our rationale behind choisng the TD3 algorithm and its variations, and finalize the section by presenting our training architecture and its implementation.

6.1 Problem definition

To pose our optimization process as a reinforcement learning problem, we must first define the markov decision process that represents the system. That is, we must define a state space representing possible observations from the environment, and an action space representing possible agent interactions with the environment. We define the state space of our Markov decision process as k-dimensional vector $\mathcal{S} : \mathbf{R}^k$ where k is the number of organs used as objectives for the transformation. For instance, $\langle 1, 1 \rangle$ may represent a crop with 1 leaf and 1 flower. Our action space is defined as an $m-$dimensional vector $\mathcal{A} : [-1, 1]^m$, with one dimension for each of the m L-System parameters, such as age and reproduction rate. As it makes training more stable and transferable [49], we opted to normalize our action space to lie between -1 and 1.

The outcome of our approach depends on the careful selection of a reward function. This is typically done through trial and error, and reward functions are typically chosen according to their domain [39]. The main difficulty in selecting a reward function lies in the extraction of a valid signal guiding the agent toward the optimal solution, as such a solution is often unknown. Thus, reward shaping the act of defining a domain-specific reward function becomes difficult. We first thought about using the negative Euclidean distance $d(s, s')$ between the observation and the target objective, but such a function would only reward the agent upon finding the optimal solution. Further, as the reward signal is unbounded, it leads to unstable learning. To make our signal strictly positive and bounded, we chose to use the dense l_2 reward.

The dense l_2 reward function is defined as $R(\cdot) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} = e^{-d(s,s')}$, where $d(s, s')$ is the Euclidean distance between states. Negative rewards obtained via the negative Euclidean distance penalize the agent when doing incorrect actions, and due to discounting, it is unclear if strictly negative rewards are conducive to learning. Negative rewards may also be interpreted differently depending on the learning algorithm. For instance, Q-learning methods only propagate to a single state [29]. By using the negative exponential of the distance, rather than the negative distance, we restrict the value of our reward function r to be $r \in [0, 1]$, where 1 signifies that an optimal solution has been found.

As a large number of solutions in our solution space are far from optimal, using the dense l_2 reward creates a sparse reward environment. That is, our agent will gain faint reward signals unless it finds near-optimal solutions. While such a reward function has proved successful for combinatorial optimization in sparse reward scenarios [24], the estimation of future discounted rewards in sparse search spaces remains an issue, as the agent is more likely to remain stuck in a local optimum. Prior work has attempted to address this issue by adapting the Bellman equation such that the policy is only updated when a new maximum reward has been found [76]. This *Max-RL* learning framework seems appropriate to our learning scenario, and knowing that the reward function plays a major role in the success of the reinforcement learning algorithm, we opted to evaluate its performance alongside a typical reward structure.

6.2 Algorithms

Successful applications of reinforcement learning typically involve experimenting with known algorithms and their hyperparameters to find a suitable function, as algorithm performance is known to be highly sensitive to their training and configuration. Thus, an agent must be trained and evaluated on a large number of experiments, which gets prohibitively expensive when simulation of the environment is costly. As such, it is of high interest to leverage algorithms that reuse data in the training process. Contrary to on-policy algorithms that continuously attempt to find an optimal action policy based on the current state of the system, off-policy algorithms train the policy independently from the actions controlling the system. Consequently, most off-policy algorithms rely

on a replay buffer, which enables the algorithm to remember actions taken on prior states of the system. As our goal is to reuse data produced by the digital twin, we focus our attention to learning algorithms with a replay buffer.

We narrowed down our algorithm selection by process of elimination. Classical Q-Learning algorithms [80] are promising for optimization-based model transformations as they can learn the optimal Q-function for any state-action pair. This Q-function evaluates the expected reward for a given state-action pair, and guides training toward optimal behavior as it is trained off-line and off-policy. However, this learning approach does not scale to continuous action spaces or high-dimensional observation spaces, which limits its application to our problem. A suitable alternative is TD3 [30], an actor-critic algorithm which learns a proxy of the Q-function off-line, and teaches an actor to perform actions through a replay buffer mechanism. Internally, the TD3 algorithm learns the Q-function by training two proxy networks, known as target networks, that regularize the algorithm's behavior over time. The critic network aims to minimize the value estimated by the targets. As the algorithm is provided by the authors of the *Max-RL* framework, TD3 seemed like a proper fit for our experiment. Thus, we implemented both the standard TD3 algorithm and its Max-RL variant. Due to prohibitive model training costs, we opted to use the same network architectures for the actor and proxy networks found in the original TD3 paper.

6.3 Training architecture and implementation

Reinforcement learning agents interact with their environment to obtain rewards and choose potential actions. To be compatible with most reinforcement learning libraries, the implementation of the environment must adhere to the Gymnasium API [71], a standard in the field. This includes defining the state and action spaces, a step function enabling the agent to progress in the environment, and a reset function that resets the environment to some initial configuration. Further, each environment must be structured in such a way that it can be contained within the vectorized environment wrapper Gymnasium provides. This wrapper enables us to instantiate multiple distinct environments that will be executed in parallel on the GPU, drastically reducing training time. To implement TD3 and TD3-Max, we based our implementation on CleanRL [36].

As the environment must be able to interact with the L-System formalism to simulate model states (i.e., concrete morphology), and do so in a parallel fashion, we had to separate the execution of the L-System from the execution of the agent training. To do so, we created a Java API that abstracts the interactions with L-Systems. Our resultant architecture can be seen on Fig. 9. To enable the interactions between our Python and Java components we developed a Python

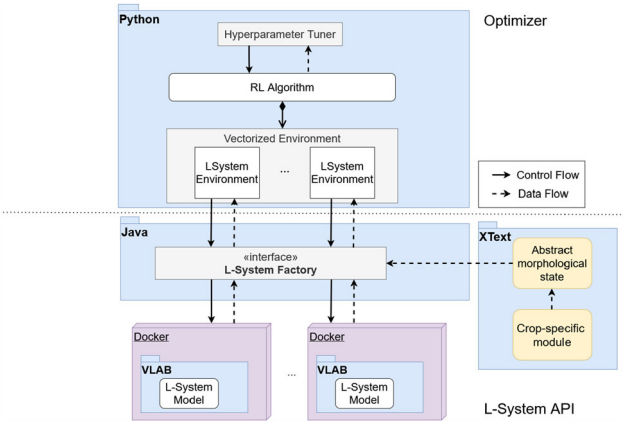

Fig. 9 Training Architecture

interface using the JPype package [61] to programmatically interact with the Java virtual machine.

The L-System formalism may be implemented with different technologies. As such, the order of application of the model's grammatical rules, the interpretation of the string, and the definition of the model will differ. For instance, VLab [64] and GroIMP [35] the most popular L-System tools rely on different data flows, languages, and structures. often writing to custom intermediary files, which complicates the execution of parallel simulations on a single machine. Specifically, VLab binaries implemented in the *LPFG* language depend on user environment variables, causing conflicts when multiple simulations are executed simultaneously. This complicates the parallel execution of string evaluations and makes integrating VLab within a reinforcement learning architecture impractical. As such, to facilitate the parallel execution of L-System models and to make our L-System interface modular, we isolate the required dependencies, environment variables, and resources within their own Docker images [57]. A Docker container is spawned every time the Python Gymnasium environment calls the L-System Java API, which parses the result and sends it back to the Python module. This isolation of processes within different containers add some computational overhead, but enable our L-System instances to be isolated, thus easily parallelizable and open to extension. Our DSL, implemented with XText [9], provides the concrete morphology optimization target to the factory and the crop-specific parameters that define how to parse the L-System string. As XText is a Java framework, it is easily integrated with our L-System API.

7 Evaluation

To investigate the potential of TD3 and TD3-Max to solve our optimization problem, we explore the refinement of straw-

berry abstract morphological states derived from an instance of phenology. We then compare the NSGA-III genetic algorithm to the TD3 implementations based on applicability, performance, and diversity criteria. We conclude this section by discussing the benefits and limitations of reinforcement learning within our methodology and how such approaches may benefit genetic algorithms.

7.1 Research objectives

Our goal is to evaluate the usage of reinforcement learning as an alternative to metaheuristics such as genetic algorithms to perform a multi-objective optimization-based model transformation. The intended purpose of integrating reinforcement learning is to enable data reuse within the digital twin. Thus, we seek to answer the following research questions (RQ):

RQ1 - Applicability: are reinforcement learning algorithms able to find a solution to our combinatorial optimization problem? In particular, can TD3 and TD3-Max find adequate solutions in a sparse reward space?
RQ2 - Performance: how do TD3 and TD3-Max compare to NSGA-III in terms of optimality and time expense?
RQ3 - Diversity: how does our reinforcement learning approach compare to NSGA-III in terms of variety of potential solutions?

To answer RQ1, we investigate the episodic distance of our reinforcement learning algorithms to assess how well our agents learned. To evaluate the performance requirements of RQ2, we compare optimal solutions found by our genetic algorithm and our learning agents, as well as wall-clock time. Given that digital twins must continuously adapt their models, time is a major factor to consider. To measure diversity and answer RQ3, we report the hypervolume metric, a normalized index of the area covered by the Pareto front.

7.2 Experimental setting

We simulate the phenology of a strawberry of the Albion cultivar using a Python simulator developed for the digital twin of our industrial partner [5]. The model outputs the crop age in days α, fruit yield $yield_{fruit}$, and organ counts for η_{fruit} fruits and η_{leaf} leaves, but fails to characterize the thyrse structure, i.e., the morphology. The experiment was conducted over a growing season of 8 weeks.

The operation of the controlled environment system depends on observable and countable attributes such as leaf and fruit counts. By optimizing the thyrse pattern, growers can effectively produce an adequate ratio of leaves to fruits

```
#define MAX_AGE 8
#define VEG_INIT 0.61
#define VEG_HALF_LIFE 0.25
#define TH_MONO 0.11
#define TH_SYM 0.8
#define TH_DIFF -0.05
#define PLASTOCHRON 0.42
#define BRACT_TH_1 0.5
#define BRACT_TH_2 0.5
```

Fig. 10 Example VLab strawberry parameter file, only optimization parameters are shown

that lead to the least energy waste as too many leaves hamper fruit yields, and too many fruits hamper the health of the crop. Thus, we must link our model of phenology to a model of morphology.

To simulate the morphological development of the crop, we use the VLab model developed by Lembinen et al. [47], which captures the dynamics of thyrse architectures of woodland strawberries. By varying the model parameters, it is possible to capture different strawberry genotypes that differ in fruit and leaf organogenesis.

In total, we vary eight L-System parameters to characterize the growth of thyrses, with 10^12 possible combinations of parameters in our morphological search space. Figure 10 showcases the parameter file used by VLab, which gets created every time the agent interacts with the environment.

The phenological model outputs the average number of fruits and leaves for the strawberry crop, so we did not need to use a translation heuristic to derive our abstract morphological state from the simulated mass. To distinguish the number of leaves between fully mature leaves (i.e., tri-lobe) and growing leaves (i.e., single-lobe), we assumed that growing conditions were adequate to support 66% of leaves to maturity. We obtain the following translation from the phenological to the abstract morphological state:

$$
\begin{aligned}
n_{fruit} &= \eta_{fruit} \\
n_{single} &= \lceil 0.33 \times \eta_{leaf} \rceil \\
n_{tri} &= \lfloor 0.66 \times \eta_{leaf} \rfloor \\
age &= \alpha
\end{aligned}
$$

The abstract state for the strawberry captures the observable measurements of interest in a vertical farming context. In our case, we care about the number of single and triple-lobed leaves, and the fruit count. The state also defines the age of the crop which we want to generate. In conformity with the metamodel, we manually apply the translation to

Table 1 Optimization results

Algorithm	Avg. solution fruit	Avg. solution single-Lobe	Avg. solution tri-Lobe	Hypervolume
NSGA-III	0 +/- 0	3 +/- 0	0 +/- 0	0.25
Random	0.11 +/- 0.25	3 +/- 0	0.86 +/- 0.42	0.24 +/- 0.014
TD3	2 +/- 0	15 +/- 0	3 +/- 0	0.25
TD3-Max	9.33 +/- 4.78	2.33 +/- 0.47	5.5 +/- 2.14	0.967

the crop simulation results via a simple python script that loads csv results and maps them to our DSL representing the abstract morphological states for our strawberry cultivars:

Listing 1 Abstract morphological states of the simulated Albion cultivar

```
Strawberry(
  SingleLobe: 3
  TriLobe: 6
  Fruits: 8
  Age: 8
)
```

7.3 Results and validation

To validate our approach, we generate the concrete morphological states of the Albion strawberry cultivar. The final simulated phenological state is exported to text and then translated into an abstract morphological state using our translation heuristic. Next, we refine the model using our transformation to generate the concrete morphological state.

7.3.1 Genetic algorithm

Genetic algorithms are stochastic and, at best, provide solutions lying on the Pareto front, which may be too computationally expensive to explore due to its size and variance. As such, we evaluate aggregate results from parallel executions by taking the mean absolute distance between the generated crop and the objective and the hypervolume, as shown in Table 1.

We parameterized the reference point set of NSGA-III by applying 15 divisions of the normalized hyperplane to preserve solution diversity. As our search space contains 10^{12} possible configurations, and as the evaluation of each solution can be computationally expensive, we kept a population size of 100 solutions evolved over 100 generations. To compare our results to random, we sampled the n best solutions generated by random, where n is the size of the NSGA-III solution set.

We observe that our approach outperforms simple random search of morphological configurations. Every execution of the genetic algorithm yielded optimal representations in terms of fruits and tri-lobe leaves, while random sometimes fails to generate such solutions. As the random algorithm still found at least one optimal configuration in almost every execution, the hypervolumes of both algorithms are nearly identical. Since our solution sets lack diversity, the hypervolume is not a good indicator to evaluate our approach; instead, we pay attention to convergence time. The results in Fig. 11 show that, on average, NSGA-III converges to an optimum in 50 generations, while random sometimes fails to converge. Convergence is an important aspect of our methodology, and we expand on that aspect in the discussion. An example of optimal strawberry representation is shown on Fig. 12.

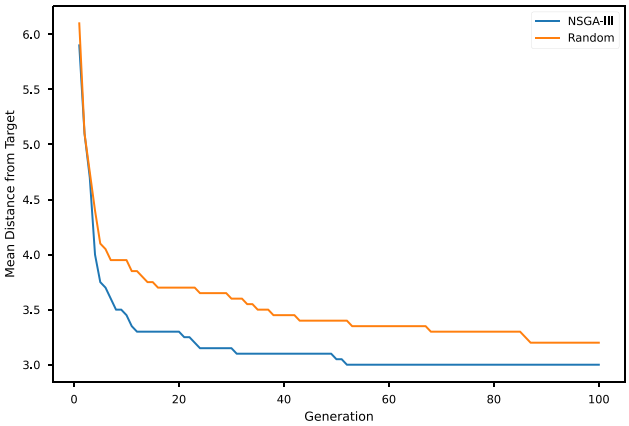

Fig. 11 Population Comparison between NSGA-III and Random



Fig. 12 Optimal morphology of the strawberry, missing 3 single-lobed leaves. Results are shown for weeks 2 to 8

7.3.2 Reinforcement learning

It is well known that the performance of machine learning algorithms is highly sensitive to their choice of hyperparameters. Due to the large possible hyperparameter space, efficient

Table 2 Hyperparameter trial configuration

Hyperparameter	Possible Values
Batch size	{1, 4, 16}
Discount Factor	{0.99, 0.9, 0.5}
Horizon	{1, 10, 20}
Learning Rate	[0.1, 0.01]
Policy Noise	{0.1, 0.2, 0.3}
Exploration Noise	{0.1, 0.2, 0.3}

techniques must be employed to calibrate the learning algorithm. One of the most common methods to efficiently calibrate such parameters is Bayesian optimization [68], which we perform using the optuna [3] Python package to find the best combination of parameters in a limited number of trials. While this does not provide an exhaustive search over the hyperparameter space and does not guarantee convergence, using such tools is common in automated machine learning [37] to make efficient use of wall-clock time. For TD3 and TD3-Max, we conducted six trials with two seeds for a total of 12 different agents. Table 2 showcases the hyperparameters that were tuned for training and their possible values, and the trial instances for TD3 and TD3-Max can be found on Tables 3 and 4, respectively. As we wish to cover more combinations of hyperparameters, the number of trained models grows exponentially and is a major limitation for experimentation. Thus, we chose possible hyperparameter values that drastically affect learning based on our knowledge of the problem, and leverage Bayesian optimization to find the best combination automatically.

While efforts have been made toward making the reinforcement learning environment parallelizable for more efficient learning in terms of wall-clock time, the L-System still takes in the order of seconds to simulate at every training iteration. As the digital twin must present actual information about the state of the system to its stakeholders, the optimization algorithm has to be re-calibrated to accommodate the evolution of the abstract morphological state. If the optimization algorithm is too slow, the digital twin will not be able to provide consistent and up-to-date views of the system. Thus, we opted to train the model for 2500 iterations, as we noticed it might take up to 8 h to finish a trial with our current hardware. Every algorithm was trained on the same machine using an NVIDIA RTX 2070 SUPER GPU.

We evaluate the ability of our agent to perform the model transformation by looking at the distance between solutions and target during training for each tuner, as seen on Fig. 13. Since the value of our reward function is very low until we achieve optimality, visualizing the value of the reward function proves challenging as it is often too close to 0. Rather, we look at the distance since it is semantically simpler yet conveys the same information. A distance of 0 indicates that the solution is optimal, and further distances indicates a worse solution. As we aim to assess whether our learning algorithms are able to solve our optimization problem by the end of training, we do not care about the cumulative sum of rewards, but rather the reward or distance obtained by the last action the agent took. Since the agent policy is deterministic, this final output should be representative of the behavior of the trained agent.

Looking at Fig. 13, we see that TD3 and TD3-Max perform differently, even accounting for the differences in hyperparameters. As TD3-Max attempts to maximize the current maximum reward rather than the sum of maximum rewards over time, the agent is much more likely to explore different actions. This behavior can be observed by looking at the variance of obtained rewards during training, as TD3-Max often explores new states compared to TD3, often leading to worse performance overall. The scale of our reward function also exemplifies the sparseness of our possible state space. Because most solutions were very far from the objective, our agent seemingly failed to converge toward optimality and hit an early reward plateau. This effect is most apparent on Tuners 2–5, where most of the rewards obtained for TD3 remained constant. We note that the best obtained solution for our reinforcement learning algorithm is outperformed by our metaheuristics, as shown in Table 5, and we expand on the topic in the discussion.

To assess the convergence of our agents, we present the behavior of the loss function for the Q-function and policy networks in Figs. 14 and 15. The loss function for the actor policy shows that TD3 and TD3-Max exhibit similar convergence properties. We observe that the actor either converges to a policy quickly or fails to learn optimal behavior, as indicated by the rising loss function after a period of negative loss values for the *Tuner 1* configuration of TD3 and *Tuner 3* for TD3-Max. Even though convergence is not guaranteed, we notice that the TD3-Max formalism is more unstable during training, as the loss for *Tuners* 2 and 4 exhibit variation whereas TD3 quickly stabilizes. This means that our actor is less prone to be stuck in a local optimum given our sparse reward scenario.

Examining the loss of our critic, we observe that TD3 and TD3-Max exhibit similar behavior. Again, the TD3-Max algorithm appears to be more unstable during learning as the loss exhibits more variation. However, unlike our actor exhibiting poor training behavior, our critic seems to converge toward optimality as the loss function converges toward 0. This indicates that our critic can match the Q-values of the target networks, meaning that the Q-function learns an accurate estimation of future rewards for arbitrary state-action pairs. However, the rapid convergence of the loss of our learned Q-function toward 0, coupled with our unreli-

Table 3 TD3 Trials

Tuner	Batch size	Discount Factor	Horizon	Learning Rate	Policy Noise	Exploration Noise	Trial execution time (h:m:s)
Tuner 1	4	0.9	20	0.0791	0.3	0.2	4:15:58
Tuner 2	1	0.9	10	0.0472	0.2	0.3	4:18:57
Tuner 3	16	0.99	1	0.0325	0.1	0.1	8:45:51
Tuner 4	16	0.9	20	0.027	0.3	0.2	3:38:35
Tuner 5	1	0.99	10	0.046	0.3	0.1	3:56:30
Tuner 6	4	0.9	1	0.0234	0.3	0.2	7:14:4

Table 4 TD3-Max Trials

Tuner	Batch size	Discount Factor	Horizon	Learning Rate	Policy Noise	Exploration Noise	Trial execution time (h:m:s)
Tuner 1	1	0.99	10	0.0391	0.2	0.1	2:53:34
Tuner 2	4	0.9	10	0.062	0.3	0.1	7:26:52
Tuner 3	1	0.9	10	0.0297	0.2	0.3	2:32:30
Tuner 4	1	0.5	10	0.048	0.3	0.2	2:33:53
Tuner 5	16	0.9	20	0.039	0.2	0.2	3:35:37
Tuner 6	16	0.9	20	0.0758	0.3	0.1	2:42:24

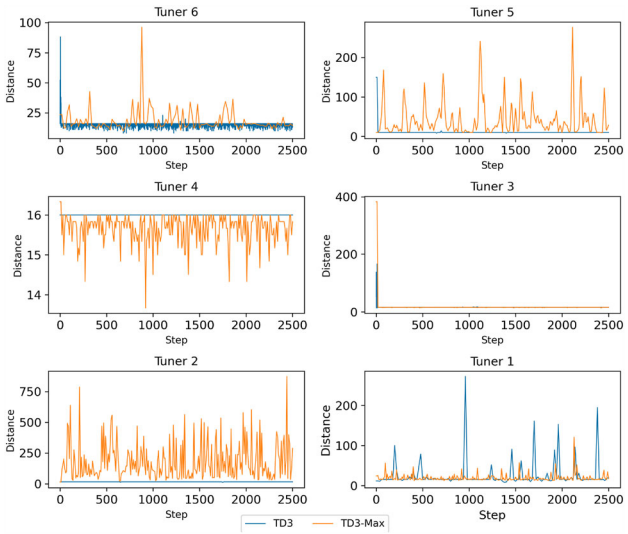

Fig. 13 Euclidian distance between solution and target for each tuner at training. Lower is better

able reward signal, indicates that the estimated Q-values are wrong. This is most likely caused by our sparse reward scenario, where true Q-values are so-close to 0 that the absolute error remains quite small.

To compare the performance of metaheuristics and reinforcement learning for model transformations, we look at the optimal solution found using each algorithm and their average runtime. We highlight the most optimal solution found using every algorithm on Table 5.

7.4 Discussion

7.4.1 Model transformation

While our approach is able to generate near-optimal representations of strawberry crops using genetic algorithms [6], the application of reinforcement learning to solve a combinatorial optimization problem remains limited. We have shown that, given a similar training time to genetic algorithms, TD3 fails to yield adequate results. Integrating learning algorithms within digital twins with such a large state space is difficult since the length of training plays a consequential role in the performance of agents. As the digital twin evolves with the system, it becomes impractical and wasteful to train a reinforcement learning algorithm over several days, given that the model will need re-tuning. However, if it were possible to efficiently train an agent over a period of a couple of hours, the model transformation could efficiently evolve along the system as we could feasibly tune its hyperparameters.

A major limitation of our approach lies in the definition of the reward function. While theoretically sound, distance-based metrics may fail to capture all the information conducive to efficient learning; thus, the agent learns much more slowly. This effect can be observed in the original TD3-Max paper [76], where the use of a dense reward function results in slower learning. Typically, reward functions must be hand-crafted and tailored to the specific problem instance. However, this usually involves some form of perfect knowledge, such as the real distance from optimal, which we do not possess in our context. Introducing some form of reward that penalizes the agent for staying in place for too

Table 5 Best algorithm solutions found through optimization

Algorithm	Avg. Runtime	Best Solution	Distance From Target
NSGA-III	4:18:12	[8, 0, 6]	[0, 3, 0]
Random	4:05:31	[8, 0, 5]	[0, 3, 1]
TD3	5:49:15	[10, 18, 3]	[2, 15, 3]
TD3-Max	3:41:50	[16, 0, 5]	[8, 0, 1]

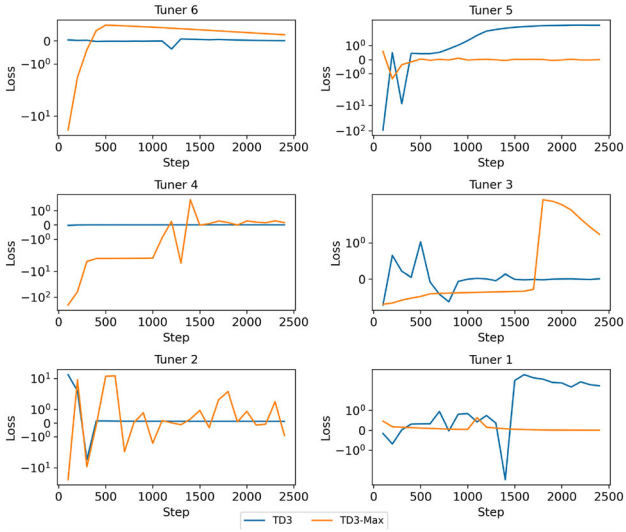

Fig. 14 Actor policy loss for each tuner

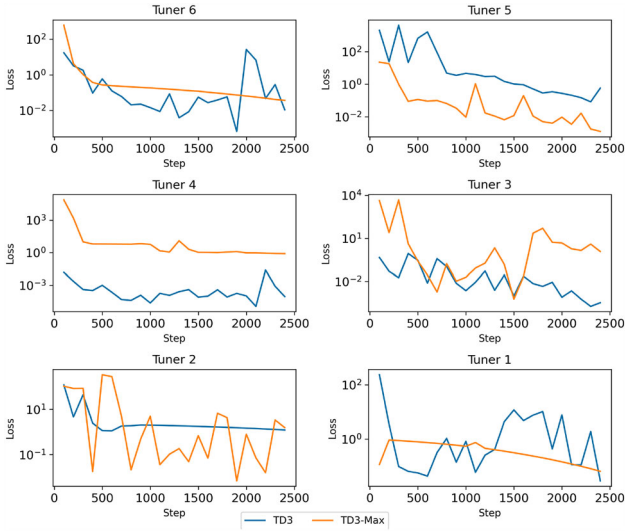

Fig. 15 Target Q-function loss for each tuner

long or avoiding costly parameter instances to simulate might be conducive to better learning performance. As the digital twin evolves, the appropriate reward function may evolve along with the system. It would be interesting to infer the reward function using the real data stream, as is done with deep generative models such as GFlowNets [40]. By doing so, it would be possible to smooth the reward function and move away from sparse reward scenarios, as is the case for the dense reward function. We can see that TD3-Max attempts to explore the state space much more, yet fails to converge. This behavior is also apparent when examining the hypervolume, for which TD3-Max has the highest. This is potentially due to the modified reward mechanism, which propagates information about past rewards and negates updating the policy if the action leads to a less optimal action. By doing so, the model is penalized for stagnating and has no incentive to stay in a local optimum.

Thus, for *RQ1* we conclude that reinforcement learning algorithms struggle to perform adequately given our optimization setting, and further investigation of algorithms and hyperparameters may yield more positive results. In the same vein, our results show that NSGA-III yields much better solutions within a similar time frame, and for *RQ2*, we conclude that metaheuristics such as NSGA-III perform better than reinforcement learning algorithms. However, we remark that TD3-Max consistently trains much faster than standard TD3, which makes it more suitable for consistent re-training. This is an important property to note, since we know the digital twin will have to re-train its agents to follow its evolution. Nonetheless, given that TD3-Max has a larger hypervolume and more variance in its results, we conclude for *RQ3* that certain reinforcement learning techniques may exhibit better diversity than NSGA-III.

Although our reinforcement learning algorithm yields worse empirical results than our metaheuristics, its ability to learn from data should not be neglected. A major limitation of our genetic algorithm optimizer is that the operators must be re-calibrated every time a new abstract morphological state is given. While this calibration process may be less sensitive than hyperparameter tuning, a lot of simulated data will be wasted as the optimization algorithm does not preserve it. In contrast, reinforcement learning algorithms have shown poorer performance in our case studies; however, they enable us to learn behaviors that potentially generalize beyond the system's operating conditions. However, training an accurate off-line algorithm requires carefully selecting hyperparameters that dictate the balance between exploration and exploitation, such as batch size or episode length, to be

practical in an operational setting. This makes the application of reinforcement learning to learn model transformations a complicated task that requires a large amount of computing power.

We believe that hybrid models mixing genetic algorithms and reinforcement learning are key to solving our optimization problem. The diversity of candidates produced by genetic algorithms such as NSGA-III [18] is far superior to solutions found via our learning algorithms. By incorporating reinforcement learning within a genetic algorithm, it could be possible to efficiently explore states around our Pareto front to detect better solutions. Another approach would be to guide the genetic algorithm calibration process with a reinforcement learning algorithm able to generalize the problem to unseen abstract morphological states. Data-driven models and methods that reuse data are key factors of the success of digital twins in smart farming [17] since data is very expensive and sometimes challenging to collect.

Nonetheless, our approach is able to generate near-optimal representations of strawberry crop morphology based on a model of their phenology, bridging the gap between operators of the vertical farm. While L-System models enforce the well-formedness of crop structure, they may use abstractions to simplify computing costs, leading to inaccurate organ counts. This lack of fidelity between crop models and their real counterparts is mitigated by the grower manually validating the proposed solutions but would require further calibration of our translation heuristic.

The generated morphologies may vary wildly in appearance and organogenesis patterns, and it is possible that no generated morphology represents the morphology of the crop in the real system. Due to this variance and the stochasticity of our genetic algorithm, we had to generate a lot of data points to have an accurate representation of possible morphologies. As digital twins continuously evolve in time, the state of the virtual system needs to be updated repeatedly. As such, the convergence speed of the optimization algorithm must be fast enough to accurately depict the system. In its current state, our methodology is fast enough to support what-if analyses but is limited in its real-time usage due to the computational cost of the L-System model evaluation.

As morphologies are not derived from sequential morphological states but rather from the initial state of the crop, our methodology is best suited to what-if analyses, where consistent and sequential morphological representations are unnecessary. The agronomist and grower can assess the desired structure of the crop and its phenological attributes to search for optimal configurations of the farming system through exploring the possible system state space, which is an integral part of prescriptive digital twins for smart farming [75].

7.4.2 MDE and consistency

Since the interactions between internal and external crop processes are not well understood in agronomy, our approach seeks to bridge the knowledge gap between agronomists and growers by transforming and integrating existing models of phenology and morphology. These models often rely on distinct formalisms and levels of abstraction, implemented across different ecosystems. To manage this heterogeneity, our model-driven approach provides a structured framework that supports model transformation and refinement, helping maintain consistency between crop representations. Using a genetic algorithm [6] we were able to generate morphologies providing the CEA digital twin a consistent view of the crop internal and external states. However, our results show that the consistency and viability of the generated morphologies highly depend on the optimization problem formulation.

By formulating the problem as an optimization task, we explicitly separate the concerns of agronomists and growers, allowing models to be adapted, reused, or exchanged across different crops and systems. A practical consequence of this separation is that, when a fitted model for a particular crop is not available, a proxy model calibrated for another crop may be reused with targeted recalibration through the heuristic. For example, if a phenological model for strawberries does not exist, one may adopt an established tomato phenology model as a proxy, then calibrate the heuristic so that the CEA Digital Twin generates morphologies and timing that are consistent with observed strawberry development. Calibration of the heuristic aligns the proxy model outputs with crop-specific traits, such as rate of leaf emergence and timing of flowering, so that the generated morphology reflects the target crop s phenology. If the model is later switched to a different crop, the same strategy applies: the computational core can be reused and only the heuristic and its calibration need to be adjusted to capture a relevant phenological state, given that the L-System model exists. This modular approach to crop modeling reduces development overhead while preserving the ability to produce biologically plausible morphologies consistent with some driving phenology, either across cultivar or species.

Consistency across formalisms is further supported by the enforcement of well-formedness constraints in L-System model structures, ensuring that simulated morphology remains compatible with phenological development of a crop when the heuristic function is well calibrated. While technical limitations in VLab prevented the generation of fully temporally consistent morphologies, since the L-System had to be reinterpreted from the initial planting date for each simulation, the optimization procedure still converged to the same solution across runs. This indicates that, even under practical constraints, the CEA Digital Twin provides a stable basis for aligning morphology and phenology. Overall,

although calibration and heuristic functions influence local accuracy, the structured modeling approach ensures that both structural and behavioral consistency are maintained, enabling exploration of the morphological state space while preserving biological plausibility. Doing so bridges the gap between expert agronomists and growers, as the interactions are implicitely modeled by the mapping of phenology onto morphology.

7.5 Threats to validity

In this section, we discuss some limits of our evaluation. We first discuss the internal validity and then report about external validity.

7.5.1 Internal validity

The successful application of reinforcement learning algorithms depends on identifying a set of hyperparameters that are suitable for the problem. However, given our limited computational resources, data availability, and the absence of an appropriate benchmark, we cannot conclude that reinforcement learning is unapplicable to the problem; instead, our experiments suggest that further work is required for its practical application. There is a lack of systematic methods that enable the successful application of reinforcement learning algorithms to various problems. Given a different network architecture and hyperparameters, our experiment may yield better results. However, this requires a significant amount of time and computational resources, and since there are no guarantees, we have opted not to test further network architectures in our experiment.

7.5.2 External validity

We have demonstrated that our simulation methodology can effectively transform an instance of crop phenology into a valid instance of morphology using our domain-specific language. While our experiment focused on strawberries, our model transformation of crop phenology to morphology is generalizable, as we can always conceive of a single or multi-objective optimization problem by extracting features from the L-System. However, this requires access to a valid phenological model of the crop, an accurate L-System model, and enough data to calibrate the translation heuristic. However, such models and data are often inaccessible due to intellectual property and business concerns, making the assessment of our methodology in different settings difficult.

8 Related work

Digital twins of cyber-biophysical systems have been applied to drug discovery [69] and livestock farming [60]. In smart agriculture, digital twin prototypes have been developed for the identification of diseases from images [74] or the harvesting of potatoes [25]. Such approaches either do not integrate the collaboration of multiple stakeholders for the operation of the system or do not model different abstractions of the physical entity. Prior work on the digital twins of a strawberry vertical farm has identified key challenges and learned lessons on the topic [17], where authors highlight the importance of domain-specific tools for digital twin operators and a push for data-driven methods.

Recent approaches in crop modeling have focused on the image-based phenomics calibration of models [15] and the extraction of morphological attributes using multiple detection methods [84]. Image-based phenomics for the inference of crop maturity stages have been applied to wheat [34] and rice [81], but the inferred phenological attributes are not sufficient to operate a vertical farm. Other approaches to bridge the gap between phenology and morphology revolve around the use of functional-structural plant models [26] (FSPM), where L-Systems are used to construct 3D representations of organs with spatiotemporal dependencies. FSPM can be computationally expensive and often do not include mechanistic models of crop development [52], as they are designed to assess crop spatial development. Each FSPM relies on its own formalism, which can be mathematically complex and typically involves the use of a specific platform, such as GreenLab [79]. Thus, FSPM do not seem suitable to support decision-making of complex farming systems.By transforming mechanistic representations of crop phenology into well-formed representations of crop morphology using MDE, we enable a separation of concerns in the digital twin that forms a base for operational decision support.

Reinforcement learning methods are gaining popularity in the smart farming domain due to their ability to capture highly complex dynamics. Recent works on smart agents in agriculture have applied transformer architectures to control an indoor strawberry farm [54] and have compared DDPG to model-based control methods to operate a lettuce greenhouse [59]. While both approaches demonstrate potential, the authors highlight the uncertainty and ad hoc nature of agent training. Given the safety risks associated with automated greenhouse control, there is a lack of formal methods to ensure the safe usage of such models within digital twins.

Smart agents have also seen recent interest in the modeling community. Work on the performance of reinforcement learning algorithms on single-objective and multi-objective model-driven optimization is in its early stage [23], and so far has only been applied to models within the same language. Approaches combining reinforcement learning to

model transformation chains have been mostly applied to the software domain, with agents for human-guided model repair [16] and for optimizing possible software transformations [22]. As shown in a recent systematic literature review [55], there is a limited corpus on the topic of reinforcement learning-based model transformations, and the trade-offs between metaheuristics, such as genetic algorithms, and data-driven methods have not been fully explored. More work is required to guide practitioners in recommending the most suitable approach for optimization-based model transformations.

9 Conclusion

In this paper, we present a methodology to generate 3D visualizations of crop morphology from a model of crop phenology for the operation of a smart agriculture prescriptive digital twin. The methodology aims to convey domain-specific representations of a crop to agronomists and growers for the operation and analysis of the vertical farm. Our proposed transformation chain enables the design space exploration of possible concrete crop morphological states on the basis of their abstract morphology, which is inferred from a phenological model. This enables the execution of what-if analyses within the digital twin, enabling the prescription of desirable crop states according to the agronomist and grower who operates the system.

We applied our methodology to the generation of strawberry crop morphologies, and have shown that we are able to transform an instance of crop phenology into crop morphology. We demonstrated that metaheuristics such as the genetic algorithm NSGA-III yield better performance and is more applicable than off-policy actor-critic reinforcement learning algorithms such as TD3, which beat NSGA-III in terms of diversity. More work is required to find ways to calibrate the methodology heuristics more precisely.

The main limitation of our methodology is its inability to model crops over horizons of time. Every new crop has to be simulated as if it were an independent entity, and future work is necessary to enable the consistent representation of crops through time. To palliate this, we aim to extend our model refinement method to take into account prior morphological states of the crop to generate results that are morphologically consistent through time. These intermediate concrete morphological states would enable the extension of our methodology to real-time monitoring of digital twins and is a step toward a prescriptive digital twin.

References

1. Abbasi, R., Martinez, P., Ahmad, R.: The digitization of agricultural industry—a systematic literature review on agriculture 4.0. Smart Agric. Technol. **2**, 100042 (2022)
2. Abelson, H., DiSessa, A.: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, Cambridge (1986)
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631 (2019)
4. Amundson, S., Deyton, D.E., Kopsell, D.A., Hitch, W., Moore, A., Sams, C.E.: Optimizing plant density and production systems to maximize yield of greenhouse-grown Trust Tomatoes. HortTechnology **22**(1), 44–48 (2012)
5. Archambault, P., David, I., Syriani, E., Sahraoui, H.: Co-Simulation for controlled environment agriculture. In: 2023 Annual Modeling and Simulation Conference (ANNSIM). SCS (2023)
6. Archambault, P., Sahraoui, H., Syriani, E.: A modeling methodology for crop representation in digital twins for smart farming. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, pp. 342–352 (2024)
7. Bergs, T., Gierlings, S., Auerbach, T., Klink, A., Schraknepper, D., Augspurger, T.: The concept of digital twin and digital shadow in manufacturing. Procedia CIRP **101**(2021), 81–84 (2021)
8. Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 (2019)
9. Bettini, L.: Implementing domain-specific languages with Xtext and Xtend. Packt Publishing Ltd (2016)
10. Bicikliski, O., Trajkova, F., Mihajlov, L.: Morphological characteristics of some pepper genotypes (*Capsicum annuum* L.) grown in conventional and organic agricultural systems: comparative analysis. Ann. Res. Rev. Biol. **28**(3), 1–11 (2018)
11. Botín-Sanabria, D.M., et al.: Digital twin technology challenges and applications: a comprehensive review. Remote Sens. **14**(6), 1335 (2022)
12. Bréda, N.J.J.: Ground-based measurements of leaf area index: a review of methods, instruments and current controversies. J. Exp. Bot. **54**(392), 2403–2417 (2003)
13. Chatzimparmpas, A., Martins, R.M., Jusufi, I., Kucher, K., Rossi, F., Kerren, A.: The state of the art in enhancing trust in machine learning models with the use of visualizations. In: Computer Graphics Forum, Vol. 39. Wiley Online Library, pp. 713–756 (2020)
14. Chmielewski, F.-M.: Phenology and Agriculture. Springer, Berlin (2003)
15. Cieslak, M, et al.: L-system models for image-based phenomics: case studies of maize and canola. in silico Plants **4**(1), diab039 (2022)
16. Dagenais, K.: Towards model repair by human opinion–guided reinforcement learning. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, pp. 192–195 (2024)
17. David, I., Archambault, P., Wolak, Q., Vu, C.V., Lalonde, T., Riaz, K., Syriani, E., Sahraoui, H.: Digital twins for cyber-biophysical systems: challenges and lessons learned. In: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE, pp. 1–12 (2023)
18. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evolut. Comput. **18**(4), 577–601 (2013)
19. Di Paola, A., Valentini, R., Santini, M.: An overview of available crop growth and yield models for studies and assessments in agriculture. J. Sci. Food Agric. **96**(3), 709–714 (2016)
20. Dieleman, J.A., De Visser, P.H.B., Meinen, E., Grit, J.G., Dueck, T.A.: Integrating morphological and physiological responses of tomato plants to light quality to the crop level by 3D modeling. Front. Plant Sci. **10**(2019), 434037 (2019)

21. Ding, W. L., Jin, H. J., Cheng, Z. J., Chen, Q.: A visualization system for tomato plant modeling. In: 2011 Eighth International Conference Computer Graphics, Imaging and Visualization, pp. 160-165. IEEE (2011)
22. Eisenberg, M., Sahay, A., Di Ruscio, D., Iovino, L., Wimmer, M., Pierantonio, A.: Multi-objective model transformation chain exploration with MOMoT. Inf. Softw. Technol. **174**(2024), 107500 (2024)
23. Eisenberg, M., Wimmer, M.: From single-objective to multi-objective reinforcement learning-based model transformation. Softw. Syst. Model. **2024**, 1–31 (2024)
24. Emami, P., Ranka, S.: Learning permutations with sinkhorn policy gradient. arXiv:1805.07010 (2018)
25. Kampker, et al.: Business models for industrial smart services-the example of a digital twin for a product-service-system for potato harvesting. Procedia Cirp **83**(2019), 534–540 (2019)
26. Evers, J.B., Van Der Werf, W., Stomph, T.J., Bastiaans, L., Anten, N.P.R.: Understanding and optimizing species mixtures using functional-structural plant modelling. J. Exp. Bot. **70**(9), 2381–2388 (2019)
27. Fass, D., Gechter, F.: Towards a theory for bio-cyber physical systems modelling. In: International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management, pp. 245–255. Springer (2015)
28. Fruth, M., Teuteberg, F.: Digitization in maritime logistics—what is there and what is missing? Cogent Bus. Manag. **4**(1), 1411066 (2017)
29. Fuchida, T., Aung, K.T., Sakuragi, A.: A study of Q-learning considering negative rewards. Artif. Life Robot. **15**(2010), 351–354 (2010)
30. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596 (2018)
31. Gomes, C., et al.: Co-simulation: state of the art. arXiv:1702.00686 (2017)
32. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: a survey. Int. J. Comput. Vis. **129**(6), 1789–1819 (2021)
33. Grieves, M., Vickers, J.: Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. Transdiscipl. Perspect. Complex Syst. New Findings Approaches **2017**, 85–113 (2017)
34. Haghshenas, A., Emam, Y.: Image-based tracking of ripening in wheat cultivar mixtures: a quantifying approach parallel to the conventional phenology. Comput. Electron. Agric. **156**(2019), 318–333 (2019)
35. Hemmerling, R., Kniemeyer, O., Lanwert, D., Kurth, W., Buck-Sorlin, G.: The rule-based language XL and the modelling environment GroIMP illustrated with simulated tree competition. Funct Plant Biol **35**(10), 739–750 (2008)
36. Huang, S., et al.: Cleanrl: high-quality single-file implementations of deep reinforcement learning algorithms. J. Mach. Learn. Res. **23**(274), 1–18 (2022)
37. Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning: Methods, Systems, Challenges. Springer, Berlin (2019)
38. Iakovidis, D., Gadanakis, Y., Campos-Gonzalez, J., Park, J.: Optimising decision support tools for the agricultural sector. Environ. Dev. Sustain. **2024**, 1–25 (2024)
39. Ibrahim, S., Mostafa, M., Jnadi, A., Salloum, H., Osinenko, P.: Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications. IEEE Access (2024)
40. Jain, M., et al.: Biological sequence design with gflownets. In: International Conference on Machine Learning. PMLR, pp. 9786–9801 (2022)
41. Jang, B., Kim, M., Harerimana, G., Kim, J.W.: Q-learning algorithms: a comprehensive classification and applications. IEEE Access **7**(2019), 133653–133667 (2019)
42. Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B.: Characterising the digital twin: a systematic literature review. CIRP J. Manuf. Sci. Technol. **29**(2020), 36–52 (2020)
43. Jones, J.W., Dayan, E., Allen, L.H., Van, Keulen, H., Challa, H.: A dynamic tomato growth and yield model (TOMGRO). Trans. ASAE **34**(2), 663–0672 (1991)
44. Jones, J.W., Hoogenboom, G., Porter, C.H., Boote, K.J., Batchelor, W.D., Hunt, L.A., Wilkens, P.W., Singh, U., Gijsman, A.J., Ritchie, J.T.: The DSSAT cropping system model. Eur. J. Agron. **18**(3–4), 235–265 (2003)
45. Kalantari, F., Tahir, O.M., Joni, R.A., Fatemi, E.: Opportunities and challenges in sustainability of vertical farming: a review. J. Landsc. Ecol. **11**(1), 35–60 (2018)
46. Lee, X.Y., Balu, A., Stoecklein, D., Ganapathysubramanian, B., Sarkar, S.: A case study of deep reinforcement learning for engineering design: Application to microfluidic devices for flow sculpting. J. Mech. Des. **141**(11), 111401 (2019)
47. Lembinen, S., Cieslak, M., Zhang, T., Mackenzie, K., Elomaa, P., Prusinkiewicz, P., Hytönen, T.: Diversity of woodland strawberry inflorescences arises from heterochrony regulated by TERMINAL FLOWER 1 and FLOWERING LOCUS T. The Plant Cell **35**(6), 2079–2094 (2023)
48. Li, D., Xu, L., Tan, C., Goodman, E.D., Fu, D., Xin, L.: Digitization and visualization of greenhouse tomato plants in indoor environments. Sensors **15**(2), 4019–4051 (2015)
49. Li, L., Lyu, J., Ma, G., Wang, Z., Yang, Z., Li, X., Li, Z.: Normalization enhances generalization in visual reinforcement learning. arXiv:2306.00656 (2023)
50. Liao, Z., et al.: Novel models for simulating maize growth based on thermal time and photothermal units: applications under various mulching practices. J. Integr. Agric. **22**(5), 1381–1395 (2023)
51. Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. J. Theor. Biol. **18**(3), 280–299 (1968)
52. Louarn, G., Song, Y.: Two decades of functional-structural plant modelling: now addressing fundamental questions in systems biology and predictive ecology. Ann. Bot. **126**(4), 501–509 (2020)
53. Lu, C., Deng, L., Fei, M.: An improved visualization modelling method of greenhouse tomato plants based on L-system. In: 2015 Chinese Automation Congress (CAC), pp. 480–485 (2015). https://doi.org/10.1109/CAC.2015.7382548
54. Lu, Y., Gong, M., Li, J., Ma, J.: Optimizing controlled environmental agriculture for strawberry cultivation using RL-Informer Model. Agronomy **13**(8), 2057 (2023)
55. Marcén, A.C., Iglesias, A., Lapeña, R., Pérez, F., Cetina, C.: A systematic literature review of model-driven engineering using machine learning. IEEE Trans. Softw. Eng. (2024)
56. Matteau, J.-P., Gumiere, S.J., Gallichand, J., Létourneau, G., Khiari, L., Gasser, M.-O., Michaud, A.: Coupling of a nitrate production model with HYDRUS to predict nitrate leaching. Agric. Water Manag. **213**(2019), 616–626 (2019)
57. Merkel, D.: Docker: lightweight Linux containers for consistent development and deployment. Linux J. **239**, 2 (2014)
58. Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M.A., Nordmoen, B., Fritzsche, M.: Where does model-driven engineering help? Experiences from three industrial cases. Softw. Syst. Model. **12**(2013), 619–639 (2013)
59. Morcego, B., Yin, W., Boersma, S., Van Henten, E., Puig, V., Sun, C.: Reinforcement learning versus model predictive control on greenhouse climate control. Comput. Electron. Agric. **215**(2023), 108372 (2023)
60. Neethirajan, S., Kemp, B.: Digital twins in livestock farming. Animals **11**(4), 1008 (2021)

61. Nelson, K.E., Scherer, M.K.: JPype. Technical Report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA (United States) (2020)
62. Pan, S.J.: Transfer learning. Learning **21**(2020), 1–2 (2020)
63. Prusinkiewicz, P.: Graphical applications of L-systems. In: Proceedings of Graphics Interface, vol. **86**, pp. 247–253 (1986)
64. Prusinkiewicz, P., Cieslak, M., Ferraro, P., Hanan, J.: Modeling plant development with L-systems. Math. Model. Plant Biol. **2018**, 139–169 (2018)
65. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, Berlin (2012)
66. Ragaveena, S., Shirly Edward, A., Surendran, U.: Smart controlled environment agriculture methods: a holistic review. Rev. Environ. Sci. Bio/Technol. **20**(4), 887–913 (2021)
67. Schumacher, A., Sihn, W., Erol, S.: Automation, digitization and digitalization and their implications for manufacturing processes. In: Innovation and Sustainability Conference Bukarest, pp. 1–5. Elsevier, Amsterdam (2016)
68. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
69. Subramanian, K.: Digital twin for drug discovery and development the virtual liver. J. Indian Inst. Sci. **100**(4), 653–662 (2020)
70. Sutton, R.S., Barto, A.G., et al.: Reinforcement learning: An introduction, vol. 1. MIT Press, Cambridge (1998)
71. Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., Arjun, K.G., et al. Gymnasium: a standard interface for reinforcement learning environments. arXiv:2407.17032 (2024)
72. Valdés, R.A., Comendador, V.F.G., Sanz, A.R., Castán, J.P.: Aviation 4.0: more safety through automation and digitization. In Aircraft technology. IntechOpen (2018)
73. Vanthoor, B.H.E., De Visser, P.H.B., Stanghellini, C., Van Henten, E.J.: A methodology for model-based greenhouse design: part 2, description and validation of a tomato yield model. Biosyst. Eng. **110**(4), 378–395 (2011)
74. Verdouw, C.N., Kruize, J.W.: Digital twins in farm management: illustrations from the FIWARE accelerators SmartAgriFood and Fractals. In: Proceedings of the 7th Asian-Australasian Conference on Precision Agriculture Digital, Hamilton, New Zealand, pp. 16–18 (2017)
75. Verdouw, C., Tekinerdogan, B., Beulens, A., Wolfert, S.: Digital twins in smart farming. Agric. Syst. **189**(2021), 103046 (2021)
76. Veviurko, G., Böhmer, W., de Weerdt, M.: To the max: reinventing reward in reinforcement learning. arXiv:2402.01361 (2024)
77. Vos, J., Evers, J.B., Hh Buck-Sorlin, G., Andrieu, B., Chelle, M., De Visser, P.H.B.: Functional-structural plant modelling: a new versatile tool in crop science. J. Exp. Bot. **61**(8), 2101–2115 (2010)
78. Wang, H., Zariphopoulou, T., Zhou, X.Y.: Reinforcement learning in continuous time and space: a stochastic control approach. J Mach Learn Res **21**(198), 1–34 (2020)
79. Wang, X., Hua, J., Kang, M., Wang, H., de Reffye, P.: Functional-structural plant model GreenLab: a state-of-the-art review. Plant Phenomics **6**(2024), 0118 (2024)
80. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**(1992), 279–292 (1992)
81. Yang, Q., Shi, L., Han, J., Yu, J., Huang, K.: A near real-time deep learning approach for detecting rice phenology based on UAV images. Agric. For. Meteorol. **287**(2020), 107938 (2020)
82. Yin, X., Struik, P.C., Goudriaan, J.: On the needs for combining physiological principles and mathematics to improve crop models. Field Crop Res **271**(2021), 108254 (2021)
83. Zekki, H., Gary, C., Gosselin, A., Gauthier, L.: Validation of a photosynthesis model through the use of the CO2 balance of a greenhouse tomato canopy. Ann. Bot. **84**(5), 591–598 (1999)
84. Zhang, Yu., Teng, P., Aono, M., Shimizu, Y., Hosoi, F., Omasa, K.: 3D monitoring for plant growth parameters in field with a single camera by multi-view approach. J. Agric. Meteorol. **74**(4), 129–139 (2018)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Pascal Archambault (MSc., Université de Montréal) is a Doctoral Candidate in the Department of Computer Science and Operations Research at Université de Montréal. His research focuses on model-driven engineering, with interest in artifical intelligence, digital twins, and agriculture. Previously, he worked on the topics of co-simulation for digital twins, where he won a best paper award at the 26th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS).



Houari Sahraoui is a professor at the software engineering lab GEODES within the Department of Computer Science and Operations Research (DIRO), Université de Montréal. He holds a Ph.D. in Computer Science (1995) from Pierre & Marie Curie University, specializing in artificial intelligence (AI). His primary research interests focus on automated software engineering (ASE) and the application of AI techniques to SE. His extensive body of work, comprising over 200 papers, has appeared in leading conferences, workshops, books, and journals. He has served as General Chair and Program Chair for numerous major conferences, including the IEEE/ACM International Conference on Automated Software Engineering (ASE), the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS), and the IEEE Working Conference on Software Visualization (VISSOFT). .



Eugene Syriani (Ph.D., McGill University, 2011) is a Full Professor in the Department of Computer Science and Operations Research at the University of Montreal. His research primarily focuses on model-driven engineering, with particular interests in domain-specific modeling languages, code generation, and collaborative modeling environments. His work also encompasses simulation-based design, the engineering of digital twins, and emerging contributions in prompt engineering with large language models. He has led multiple research projects in Canada that explore the application of these topics across various domains. He has an extensive history of teaching software modeling and design courses. He has directed the development of publicly available software frameworks and tools to enhance the productivity of domain experts and emphasize user experience..