Learning crop management by reinforcement: gym-DSSAT

Romain Gautron, Emilio J Padrón, Philippe Preux, Julien Bigot, Odalric-Ambrym Maillard, Gerrit Hoogenboom, Julien Teigny

Learning Crop Management by Reinforcement: gym-DSSAT

Romain Gautron, [1,2] Emilio J. Padrón, [3] Philippe Preux, [4] Julien Bigot, [5] Odalric-Ambrym Maillard, [6] Gerrit Hoogenboom, [7] Julien Teigny [6]
[1]AIDA, Univ Montpellier, CIRAD, Montpellier, France
[2]CGIAR Platform for Big Data in Agriculture, Alliance of Bioversity International and CIAT, Cali-Palmira 763537, Colombia
[3]UDC–Computer Architecture Group & CITIC & Edif. Área Científica, Campus Elviña S/N 15071, A Coruña, Spain
[4]Université de Lille, CNRS, Inria, F-59650 Villeneuve d'Ascq, France
[5]Université Paris-Saclay, UVSQ, CNRS, CEA, Maison de la Simulation, 91191, Gif-sur-Yvette, France
[6]Université de Lille, Inria, CNRS, Centrale Lille UMR 9189 – CRIStAL, F-59000 Lille, France
[7]Agricultural and Biological Engineering 289 Frazier Rogers Hall University of Florida Gainesville, Florida 32611-0570, USA
r.gautron@cgiar.org, emilio.padron@udc.gal, philippe.preux@inria.fr, julien.bigot@cea.fr, odalric.maillard@inria.fr, gerrit@ufl.edu, julien.teigny@inria.fr

Abstract

We introduce gym-DSSAT, a gym environment for crop management tasks, that is easy to use for training Reinforcement Learning (RL) agents. gym-DSSAT is based on DSSAT, a state-of-the-art mechanistic crop growth simulator. We modify DSSAT so that an external software agent can interact with it to control the actions performed in a crop field during a growing season. The RL environment provides predefined decision problems without having to manipulate the complex crop simulator. We report encouraging preliminary results on a use case of nitrogen fertilization for maize. This work opens up opportunities to explore new sustainable crop management strategies with RL, and provides RL researchers with an original set of challenging tasks to investigate.

Introduction

During a growing season, a farmer performs a sequence of operations in her field in order to reach certain production objectives (Sebillotte 1974, 1978). She makes these decisions under uncertainty, like unknown weather changes. Reinforcement Learning (RL) addresses such problems where an agent learns to control the evolution of an unknown and uncertain dynamical system, in order to perform a given task (Sutton and Barto 2018). In RL, addressing a complex real-world problem usually starts with the use of a high-fidelity simulator which mimics real learning conditions. We present gym-DSSAT, an RL environment based on a celebrated high-fidelity crop model, the Decision Support System for Agrotechnology Transfer (DSSAT, Hoogenboom et al. 2019) crop model. Learning sustainable crop management practices is not a trivial task. For example, nitrogen fertilization requires minimal rainfall and temperature following the application for the fertilized nitrogen to become available to plants. Future meteorological conditions are not known with certainty at the time of fertilization decisions. For an efficient nitrogen fertilizer management, available nitrogen in soil must match plant uptake, both in time and quantity (Meisinger and Delgado 2002). RL is an appealing approach to help decision-makers to learn more sustainable crop management practices (Binas, Luginbuehl, and Bengio 2019; Gautron et al. 2022a)

Contributions. We introduce gym-DSSAT, a crop management simulator to be used to train RL agents based on the DSSAT crop model system. gym-DSSAT features three predefined problems. We provide preliminary experimental results indicating that RL is an interesting way to discover original and efficient crop management strategies. As another contribution, Gautron et al. (2022b) details the original methods that we designed to turn DSSAT –a large mechanistic model written in Fortran–, into a Python gym environment (not discussed in this article). More information is available on gym-DSSAT GitLab[1], including installation instructions for various operating systems, and tutorials.

Related works. The first case of an RL agent interacting with a crop simulator in order to learn crop management is found in Garcia (1999). The author used a modification of the Déciblé crop model (Chatelin et al. 2005). The RL agent learned wheat sowing and nitrogen fertilization under pollution constraints. During simulations, weather series were stochastically generated. The modified version of Déciblé is not available anymore. In Garcia (1999), the RL agent did not outperform the crop management policy of an expert. Recently, several works directly used crop models or surrogate models as RL environments (e.g. Sun et al. 2017; Wang, He, and Luo 2020; Chen et al. 2021). However, none of these works has provided an open source and standardized crop management RL environment. Overweg, Berghuijs, and Athanasiadis (2021) proposed CropGym, a gym interface to train an agent to perform wheat nitrogen fertilization. The environment uses the Python Crop Simulation Environment (PCSE) LINTUL3 (Shibu et al. 2010) wheat crop model. Fertilization is treated as a weekly choice of a discrete amount of fertilizer to apply. In CropGym, simulations use a limited set of historical weather records, which may favor overfitting due to limited randomness, compared with the use of a stochastic weather generator, especially for data intensive algorithms used in deep RL.

[1]Repository: https://gitlab.inria.fr/rgautron/gym_dssat_pdi

Formalizing decision-making problems in RL

In most cases, RL uses a Markov Decision Process (MDP, Puterman 1994) formulation of the environment. An MDP defines a class of controllable dynamical system. An agent learns to control the system to optimize a certain objective function J. At each discrete time step $t \in \{1, 2, \cdots, N\}, N \leq \infty$, the system is in some state $s_t \in \mathcal{S}$ in which one action a_t from a set of actions $\mathcal{A}$ is performed by the agent. Then, the system transits into its next state s_{t+1} according to a transition function $\mathbf{p}(s, a, s')$, which specifies the probability of the system to transit to state s' after action a was performed in state s. After an action a_t has been performed, a return $r_t \in \mathbb{R}$ is provided to the agent according to the return function $\mathbf{r}(s, a, s')$. *The goal of an RL agent is to learn an optimal policy $\pi^\star(s)$ that specifies which action should be performed in each state, in order to optimize J.* For example, when $N < \infty$, the objective function can be defined as the sum of returns: $J = \sum_{t=1}^{N} r_t$. In RL, neither $\mathbf{p}$ nor $\mathbf{r}$ is known. The agent learns an optimal policy by interacting with its environment, i.e. the dynamical system to control. The agent tries actions to learn their consequences and, progressively, focuses on the best actions to perform to maximize J. Current state-of-the-art RL algorithms are known to be actor-critics, such as PPO, A2C and SAC (Kiran et al. 2021).

`gym` environments. OpenAI `gym` (Brockman et al. 2016) is an open source toolkit initially developed by the Open AI company. It provides light RL environments with a standardized Application Programming Interface (API). `gym` API became a reference in the RL community to create standardized RL environments in order to compare performances of RL algorithms. The user interacts with the environment through standardized methods. The agent interacts with the environment by calling the `step()` method with argument a_t specifying the action to take, in order to receive s_{t+1} and r_t. The objective function J is defined by the user.

Crop management problems in `gym-DSSAT`

DSSAT is meant to simulate one crop field during one growing season. The growth of one cultivar is simulated per unit of surface. Hence, the whole area of the field is meaningless, and the properties of the soil and the weather conditions are supposed to be identical in the whole field. Initial conditions may be set to simulate crop rotation along seasons. By default, `gym-DSSAT` simulates a maize experiment which has been carried out in 1982 in the experimental farm of the University of Florida, Gainesville, USA (Bennett et al. 1989). An episode lasts a simulated growing season. A simulation starts prior to planting and ends at crop harvest, which is automatically defined as the crop maturity date. Crop maturity depends on crop growth, which depends itself on crop management and weather events, and the time to reach it is stochastic. During a growing season (160 days on average), an RL agent daily decides on the crop management action(s) to perform: fertilize and/or irrigate. By default, for each episode, the weather is generated by the `WGEN` stochastic weather simulator (Richardson 1985; Soltani and Hoogenboom 2003).The duration between the

Action	Description	Range
fertilization	nitrogen amount (kg/ha)	[0,200]
irrigation	water amount (l/m^2)	[0,50]

Table 1: Daily actions available in `gym-DSSAT`

Day After Planting (DAP)	Quantity (kg N/ha)
40	27
45	35
80	54

Table 2: Expert fertilization policy

starting date of the simulation and the planting date, which lasts about one month, induces stochastic soil conditions at the time of planting (e.g. soil nitrate, and soil water content), as a result of the stochastic weather events. The number of measurable attributes in a field is extremely large (e.g. Husson et al. 2021). Based on agronomic knowledge, we selected a subset of `DSSAT` state variables with the constraint that these variables are measurable or can be estimated in real conditions. These observation variables are mixed, and take either continuous or discrete values. In `DSSAT`, the `WGEN` stochastic weather simulator is implemented as a first-order Markov chain, but all other processes are deterministic. Therefore, `gym-DSSAT` decision problems are Markovian. Because the agent only accesses a subset of all `DSSAT` internal variables, a `gym-DSSAT` problem is a Partially Observable MDP (POMDP, Åström 1965), a situation alike the one faced by farmers. In contrast with many RL toy environments, the environment is autonomous: it evolves by itself and not only because an action has been performed by the agent. Indeed, if on a given day a farmer does not fertilize/irrigate, the plot still evolves. `DSSAT` simulates the dynamics at the plot level. Likewise, the agent performs actions on the whole plot. Growing conditions such as soil characteristics and other crop operations such as soil tillage and cultivar choices are fixed.

By default, `gym-DSSAT` provides three RL problems:

1. A **fertilization problem** in which the agent can apply every day a certain quantity of nitrogen (Table 1). Crops are rainfed, and no irrigation is applied during the growing season, except a single one before planting. We crafted the default fertilization return function as:

$$r(t) = \underbrace{\texttt{trnu}(t, t+1)}_{\substack{\text{plant nitrogen} \\ \text{uptake (kg/ha)}}} - \underbrace{0.5}_{\substack{\text{penalty} \\ \text{factor}}} \times \underbrace{\texttt{anfer}(t)}_{\substack{\text{fertilizer} \\ \text{quantity (kg/ha)}}} \quad (1)$$

2. An **irrigation problem** in which the agent can provide every day a certain amount of water to irrigate, as indicated in Table 1. Independently of these irrigation actions, nitrogen fertilization occurs following the schedule provided in Table 2.
3. A mixed **fertilization and irrigation problem** which combines both the aforementioned decision problems, i.e. the agent can fertilize and/or irrigate every day.

Variable	Definition
`istage`	`DSSAT` maize growing stage (categorical)
`vstage`	vegetative growth stage (number of leaves)
`topwt`	above the ground crop biomass (kg/ha)
`grnwt`	grain weight dry matter (kg/ha)
`swfac`	index of plant water stress (unitless)
`nstres`	index of plant nitrogen stress (unitless)
`xlai`	leaf area index (m^2 leaf/m^2 soil)
`dtt`	growing degree days ($^\circ$C.day)
`dap`	days after planting (day)
`cumsumfert`	cumulative nitrogen fertilization (kg N/ha)
`rain`	rainfall for the current day (l/m^2/day)
`ep`	actual plant transpiration rate (l/m^2/day)

Table 3: Default observation space for the fertilization task

Custom scenario definition. A user can easily modify the observation space in the YAML configuration file. In the same way, the definition of the return functions can be easily modified by the user by editing a standalone Python file. Built-in `DSSAT` features can be directly leveraged, such as environmental modifications with changes in atmospheric CO_2 concentration or meteorological features, to mimic the effects of climate change. It is also possible to customize the soil conditions of the crop field, its initial conditions, as well as the weather along the season either reproducing the precipitations from a dataset, or by using the WGEN simulator.

Experimenting with `gym-DSSAT`

A use case: learning an efficient maize fertilization

We present *an example of how to address the default fertilization task*. The source code of these experiments is available in `gym-DSSAT` GitLab. We used `gym-DSSAT v0.0.7`.

Methods. An episode spans one growing season, i.e. a finite number of time steps, we define the objective function as the sum of returns J. As a common practice, we pragmatically approximate this decision problem as an MDP, even though it is a POMDP. Table 3 provides the observation space. We consider three policies:

- *The "null" policy that never fertilizes*. As there is always nitrogen in soil before cultivation (Morris et al. 2018), without mineral fertilization, the reference experiment, or *control*, is usually the null policy. Agronomists measure the effect of a nitrogen fertilization policy as a gain compared to the null policy, in order to decouple the effect of nitrogen fertilization from the effect of already available nitrogen in soil (Vanlauwe et al. 2011).
- *An "expert" policy published in the original maize field experiment* (Bennett et al. 1989) and defined in Table 2. This expert policy consists of three deterministic nitrogen fertilizer applications, which only depends on the number of days after planting.
- *A policy learned by the Proximal Policy Optimization* (PPO, Schulman et al. 2017) RL algorithm, as implemented in `Stable-Baselines3 1.4.0` (Hill et al. 2018). As our goal is to establish a simple baseline, we use the default hyper-parameter values for PPO. We trained PPO during 10^6 episodes. During training, the performance of PPO is evaluated on a validation environment every 10^3 episodes. We seed the validation environment with a different seed than for the training environment. Consequently, the validation environment generates a different sequence of weather series compared to the training environment. The model with the best validation performance is saved as the result of the training.

In order to compare fertilization policies, we measure their performances by running them for 10^3 episodes on a test environment. With regards to the training environment, the test environment is the same except for the seed of the pseudo-random number generator. In the performance analysis of policies, the evolution of returns r_t provides information about the learning process from an RL perspective, but returns are not directly interpretable from an agronomic perspective. Performance analysis of crop management strategies require multiple evaluation criteria (Doré et al. 2006; Duru et al. 2015). To remedy this problem, we use a subset of `DSSAT` internal state variables as performance indicators (Table 4). Note that these variables are not necessarily contained in the observation space of the fertilization problem (Table 3) because we use them for another purpose than algorithm training. Each of these performance criteria is correlated with the other ones. For instance, increasing the total fertilizer amount is likely to increase the grain yield, but it is also likely to increase the pollution induced by nitrate leaching. The agronomic nitrogen-use efficiency (ANE, Vanlauwe et al. 2011) is a common indicator of fertilization sustainability. For a policy π, let $\texttt{grnwt}^\pi$ be the dry matter grain yield of the policy π (kg/ha), $\texttt{grnwt}^0$ be the dry matter grain yield with no fertilization (kg/ha), and $\texttt{cumsumfert}^\pi$ be the total fertilizer quantity applied with policy π (kg/ha), we have:

$$\mathrm{ANE}^\pi(t) = \frac{\texttt{grnwt}^\pi(t) - \texttt{grnwt}^0(t)}{\texttt{cumsumfert}^\pi(t)} \tag{2}$$

ANE indicates the grain yield response with respect to the null policy provided by each unit of nitrogen fertilizer. ANE is a key metric of sustainable fertilization. Maximizing ANE relates to the economic and environmental aspects, and leads to an efficient use of fertilizer, which limits the risks of pollution. Performance indicators listed in Table 4 show a complex trade-off between conflicting objectives.

Results. Figure 1 illustrates the evolution of the objective function J against the day of simulation. PPO outperforms the two other policies. The performance obtained by PPO learned policy is less variable than that of the expert policy. Figure 2 provides a 2D histogram of fertilizer applications, against the day of simulation. PPO nitrogen fertilizer applications are more frequent at the beginning of the growing season and around day of simulation 60. This date corresponds to the beginning of the floral initiation stage. Nevertheless, *the variability of rates and application dates of PPO policy shows that it does not depend solely on the number of days after planting as the expert policy, but also depends on other factors*. Table 5 provides statistics of the performance

Variable	Definition	Comment
`grnwt`	grain yield (kg/ha)	quantitative objective to be maximized
`pcngrn`	nitrogen content in grains (%)	qualitative objective to be maximized
`cumsumfert`	total fertilization (kg/ha)	cost to be minimized
–	application number	cost to be minimized
–	nitrogen use efficiency (kg/kg)	agronomic criteria to be maximized
`cleach`	nitrate leaching (kg/ha)	loss/pollution to be minimized

Table 4: Performance indicators for fertilization policies. '-' means the variable is not provided by default but it can be derived.

	Null	Expert	PPO
grain yield (kg/ha)	1141.1 (344.0)	**3686.5** (1841.0)	3463.1 (1628.4)
massic nitrogen in grains (%)	1.1 (0.1)	**1.7** (0.2)	1.5 (0.3)
total fertilization (kg/ha)	**0** (0)	115.8 (5.2)	82.8 (15.2)
application number	**0** (0)	3.0 (0.1)	5.7 (1.6)
nitrogen use efficiency (kg/kg)	n.a.	22.0 (14.1)	**28.3** (16.7)
nitrate leaching (kg/ha)	**15.9** (7.7)	18.0 (12.0)	18.3 (11.6)

Table 5: Mean (st. dev.) of performances computed over 1000 episodes. **Bold** numbers indicate the best performing policy.

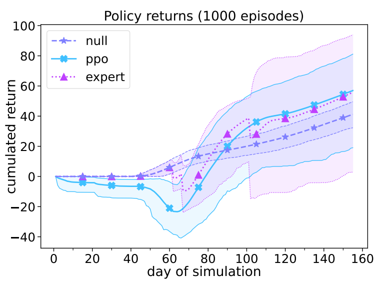

Figure 1: Mean cumulated return of each of the 3 policies against the day of simulation. Shaded area displays the $[0.05, 0.95]$ quantile range for each policy.

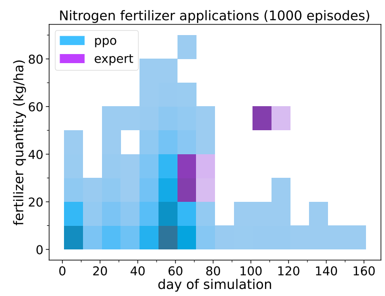

Figure 2: 2D histogram of fertilizer applications (the darker the more frequent).

indicators mentioned in Table 4. As expected, no policy is optimal for all the performance criteria. PPO policy exhibits a good performance trade-off between the expert and the null policies. Grain yield and nitrogen content in grains (a nutritional criteria) are close to those of the expert policy. On average, PPO policy consumes about 28% less nitrogen than the expert policy. Consistently, ANE for PPO is about 29% larger than that of the expert policy. From a practical perspective, a good fertilization policy consists of a limited number of applications of the fertilizer as the expert policy suggests. Indeed, each application costs in terms of fertilizer and its application. The mean number of applications of PPO (~ 6) is larger than for the expert policy (3) but still remains manageable.

Execution time. We performed all the experiments with `gym-DSSAT` on a standard 8-core laptop. The mean running time to simulate one day in `gym-DSSAT`, i.e. taking a single step in the environment, is 2.56 ± 0.22 ms. Thus, each interaction is fast and this allows a large number of interactions to be considered for training the agent.

Reproducibility. We successfully reproduced the results of this study on the same hardware and software layers. This means that both results of `gym-DSSAT` and `Stable-Baselines3` PPO are reproducible on the same platform. Nevertheless, as a more general reproducibility issue, we cannot guarantee the cross-platform reproducibility of the experiments that we presented. If we consider only `gym-DSSAT`, we have successfully reproduced the outputs of the environment across various Linux platforms.

Concluding remarks

We presented `gym-DSSAT`, *a gym environment to train RL agents for realistic crop management tasks*. `gym-DSSAT` provides the RL community with a state-of-the-art crop simulator that features original challenges. The preliminary results, which we present here, confirm that, in simulated conditions, RL can discover interesting crop management policies. `gym-DSSAT` also allows world-wide growing conditions to be mimicked, using already widely available `DSSAT` simulation files. `gym-DSSAT` can be an important tool for addressing the ongoing challenges of sustainable crop intensification through improved crop management, including those in the Global South.

Acknowledgments

The authors acknowledge the PDI team, in particular Karol Sierocinski, for their help. They also thank the DSSAT development team, especially Cheryl Porter for her continuous support. We acknowledge the Consultative Group for International Agricultural Research (CGIAR) Platform for Big Data in Agriculture and we especially thank Brian King. Ph. Preux, and O-A. Maillard acknowledge the support of the Métropole Européenne de Lille (MEL), ANR, Inria, Université de Lille, through the AI chair Apprenf number R-PILOTE-19-004-APPRENF. We acknowledge the support of the AIDA team at CIRAD and the outstanding working environment provided by Inria in the Scool research group. Emilio J. Padrón's work was partially supported through the research projects PID2019-104184RB-I00 funded by MCIN/AEI/10.13039/501100011033, and ED431C 2021/30, ED431F 2021/11 and ED431G 2019/01 funded by Xunta de Galicia. We thank Debabrota Basu for his help in the writing of this paper.

References

Åström, K. J. 1965. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1): 174–205.

Bennett, J.; Mutti, L.; Rao, P.; and Jones, J. 1989. Interactive effects of nitrogen and water stresses on biomass accumulation, nitrogen uptake, and seed yield of maize. *Field Crops Research*, 19(4): 297–311.

Binas, J.; Luginbuehl, L.; and Bengio, Y. 2019. Reinforcement Learning for Sustainable Agriculture. In *ICML Workshop Climate Change: How Can AI Help?*

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chatelin, M.-H.; Aubry, C.; Poussin, J.-C.; Meynard, J.-M.; Massé, J.; Verjux, N.; Gate, P.; and Le Bris, X. 2005. DéciBlé, a software package for wheat crop management simulation. *Agricultural Systems*, 83(1): 77–99.

Chen, M.; Cui, Y.; Wang, X.; Xie, H.; Liu, F.; Luo, T.; Zheng, S.; and Luo, Y. 2021. A reinforcement learning approach to irrigation decision-making for rice using weather forecasts. *Agricultural Water Management*, 250: 106838.

Doré, T.; Martin, P.; Le Bail, M.; Ney, B.; and Roger-Estrade, J. 2006. *L'agronomie aujourd'hui*. Editions Quae.

Duru, M.; Therond, O.; Martin, G.; Martin-Clouaire, R.; Magne, M.-A.; Justes, E.; Journet, E.-P.; Aubertot, J.-N.; Savary, S.; Bergez, J.-E.; et al. 2015. How to implement biodiversity-based agriculture to enhance ecosystem services: a review. *Agronomy for sustainable development*, 35(4): 1259–1281.

Garcia, F. 1999. Use of reinforcement learning and simulation to optimize wheat crop technical management. In *Proceedings of the International Congress on Modelling and Simulation (MODSIM'99)*, 801–806.

Gautron, R.; Maillard, O.-A.; Preux, P.; Corbeels, M.; and Sabbadin, R. 2022a. Reinforcement learning for crop management support: Review, prospects and challenges. *Computers and Electronics in Agriculture*, 200: 107182.

Gautron, R.; Padrón, E. J.; Preux, P.; Bigot, J.; Maillard, O.-A.; and Emukpere, D. 2022b. gym-DSSAT: a crop model turned into a Reinforcement Learning environment. Research Report RR-9460, Inria Lille.

Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

Hoogenboom, G.; Porter, C.; Boote, K.; Shelia, V.; Wilkens, P.; Singh, U.; White, J.; Asseng, S.; Lizaso, J.; Moreno, L.; et al. 2019. The DSSAT crop modeling ecosystem. *Advances in crop modelling for a sustainable agriculture*, 173–216.

Husson, O.; Sarthou, J.-P.; Bousset, L.; Ratnadass, A.; Schmidt, H.-P.; Kempf, J.; Husson, B.; Tingry, S.; Aubertot, J.-N.; Deguine, J.-P.; Goebel, F.-R.; and Lamichhane, J. R. 2021. Soil and plant health in relation to dynamic sustainment of Eh and pH homeostasis: A review. *Plant and Soil*.

Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

Meisinger, J. J.; and Delgado, J. A. 2002. Principles for managing nitrogen leaching. *Journal of soil and water conservation*, 57(6): 485–498.

Morris, T. F.; Murrell, T. S.; Beegle, D. B.; Camberato, J. J.; Ferguson, R. B.; Grove, J.; Ketterings, Q.; Kyveryga, P. M.; Laboski, C. A.; McGrath, J. M.; et al. 2018. Strengths and limitations of nitrogen rate recommendations for corn and opportunities for improvement. *Agronomy Journal*, 110(1): 1.

Overweg, H.; Berghuijs, H. N.; and Athanasiadis, I. N. 2021. CropGym: a Reinforcement Learning Environment for Crop Management. arXiv:2104.04326.

Puterman, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Richardson, C. 1985. Weather simulation for crop management models. *Transactions of the ASAE*, 28(5): 1602–1606.

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

Sebillotte, M. 1974. Agronomie et agriculture. Essai d'analyse des tâches de l'agronome. *Cahiers Orstom, série biologie*, 24: 3–25.

Sebillotte, M. 1978. Itinéraires techniques et évolution de la pensée agronomique. *CR Acad. Agric. Fr*, 64(11): 906–914.

Shibu, M. E.; Leffelaar, P. A.; Van Keulen, H.; and Aggarwal, P. K. 2010. LINTUL3, a simulation model for nitrogen-limited situations: Application to rice. *European Journal of Agronomy*, 32(4): 255–271.

Soltani, A.; and Hoogenboom, G. 2003. A statistical comparison of the stochastic weather generators WGEN and SIMMETEO. *Climate Research*, 24(3): 215–230.

Sun, L.; Yang, Y.; Hu, J.; Porter, D.; Marek, T.; and Hillyer, C. 2017. Reinforcement learning control for water-efficient agricultural irrigation. In *Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 1334–1341. IEEE.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Vanlauwe, B.; Kihara, J.; Chivenge, P.; Pypers, P.; Coe, R.; and Six, J. 2011. Agronomic use efficiency of N fertilizer in maize-based systems in sub-Saharan Africa within the context of integrated soil fertility management. *Plant and soil*, 339(1): 35–50.

Wang, L.; He, X.; and Luo, D. 2020. Deep reinforcement learning for greenhouse climate control. In *Proceedings of the 2020 IEEE International Conference on Knowledge Graph (ICKG)*, 474–480. IEEE.