Deep Reinforcement Learning for Greenhouse Climate Control

Lu Wang[1], Xiaofeng He[1*], Dijun Luo[2*]
[1] School of Computer Science and Technology, East China Normal University
[2] Tencent AI lab, Inc.
[1]luwang@stu.ecnu.edu.cn, xfhe@sei.ecnu.edu.cn, [2]dijunluo@tencent.com

***Abstract*—Worldwide, the area of greenhouse production is increasing with the rapid growth of global population and demands for fresh food. However, the greenhouse industry encounters challenges to find automatic control policy. Reinforcement Learning (RL) is a powerful tool in solving the autonomous decision making problems. In this paper, we propose a novel Deep Reinforcement Learning framework for cucumber climate control. Although some machine learning methods have been proposed to address the dynamic climate control problem, these methods have two major issues. First, they only consider the current reward (e.g., the fruit weight of the cucumber). Second, previous study only considers one control variable. However, the growth of crops are impacted by multiple factors synchronously (e.g., CO2 and Temperature).To solve these challenges, we propose a Deep Reinforcement learning based climate control method, which can model future reward explicitly. We further consider the fruit weight and the cost of the planting in order to improve the cumulative fruit weight and reduce the costs.**

Extensive experiments are conducted on the cucumber simulator environment have shown the superior performance of our methods.

***Index Terms*—On policy Reinforcement Learning; Cucumber Climate Control.**

I. Introduction

With the increasing of the rapid growth of global population, the demand for healthy fresh food is rising as well ([1]). The greenhouse industry plays an important role in providing the fresh food with high vitamins and minerals. Greenhouses allow a high crop production and low resource cost with well control skill ([2]). Worldwide, the area of greenhouse production is increasing ([3]). However, the greenhouse industry encounters difficulties in finding enough skilled labors to manage the crop production. The most standard way to control the greenhouse is based on the crop managers with high level of knowledge and experience.

An effective greenhouse requires the system to actively control the states of the actuators, for example, lighting, irrigation and ventilation to produce a desirable growing climate. The early autonomous greenhouse control strategy requires the grower to set the heating, lighting and etc. setpoints beforehand. After that, the actuators are operated follow these setpoints.

The climate control process is an Markov decision process, where the current decision or action (for example irrigation amount, CO_2 concentration) will affect the transition state (for example, the height of the cucumber). To add more autonomous control, a set of research works focus on designing a greenhouse model and utilize dynamic programming methods to learn the control policy ([4]–[6]). However, these methods learn a policy by interacting with the environment or greenhouse simulator, which causes the learned policy highly count on the greenhouse model. In addition, these dynamic programming methods can only control discrete state space and one dimension action space.

Indeed a set of planting trajectories which is generated by the behavior policy (for example, the human grower's policy) can be saved by the greenhouse, which inspired us to utilize AI algorithms to learn an automatic policy. However, in the greenhouse autonomous control problem, collecting the dataset from the real world is extremely costly and risky. Given these limited set of pre-collected trajectories, non-optimal behavior data, imitation learning is unable to learn a target policy. In addition, it fails in sub-optimal behavior policy datasets ([7], [8]). The success of reinforcement learning is to learn a target policy via self-interacting with the environment to solve the MDP problem, which mostly relies on an experience replay buffer collected by the growing trajectory data via interacting. But it is incapable of learning with uncorrelated data due to the gap between the behavior policy and the learned policy.

This gap rises a problem where unseen state-action pairs are erroneously estimated to have unrealistic values. While the distribution between the behavior policies and the target policy are mismatched, it would unable to estimate the target whose actions are not contained in the training data.

For effectively utilizing the fixed dataset, we first model the distribution of the actions taken by the behavior policies via a generative model followed by optimizing the target policy via traditional reinforcement learning algorithms and the reward signal of the behavior policy. Due to the period control property of planting cucumber, we introduce a period variational auto-encoder (VAE) model ([9]) to produce the periodic control policy.

Our contributions can be summarized as follows:

- To our best knowledge, it is the first work to utilize deep reinforcement learning on learning a continuous autonomous climate control policy.
- We proposed CPRL to learn a policy without interacting with the environment via two-step: (1) matching the distribution between behavior policy trajectories and (2)

*Corresponding authors.

978-1-7281-8156-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICBK50248.2020.00073

optimizing the target policy via the behavior policy's reward signal.
- Experiments on the a greenhouse simulator demonstrated that CPRL achieves the state-of-the-art performance.

II. Background

In this section, we give a definition of the Dynamic Treatment Regime (DTR) problem and an overview of preliminaries for our approach. Some important notations mentioned in this paper are summarized in Table I.

A. Problem Formulation

In this paper, DTR is modeled as a *Markov decision process* (MDP) with finite time steps and a deterministic policy consisting of an action space $\mathcal{A}$, a state space $\mathcal{S}$, and a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. At each time step t, a doctor observes the current state $s_t \in \mathcal{S}$ of a patient, chooses the medication $\hat{a}_t \in \mathbb{R}^K$ from candidate set $\mathcal{A}$ based on an unknown policy $\hat{\mu}(s)$, and receives a reward r_t. Our goal is to learn a policy $\mu_\theta(s)$ to select an action (medication) a_t which maximizes the sum of discounted reward (return) from time step t, which is defined as $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$, and simultaneously minimizes the difference from clinician decisions $\hat{a}_t$.

There are two types of methods to learn the policy: (1) value-based RL learns a greedy policy $\mu(s)$ and (2) policy gradient RL maintains a parameterized stochastic policy $\pi_\theta(a|s)$ or a deterministic policy $\mu_\theta(s)$ (see II-B for details), where θ are the parameters of the policy.

B. Model Preliminaries

The off-policy method Q-learning [**?**] learns a greedy policy $\mu(s) = argmax_a Q^\mu(s, a)$, where $Q^\mu(s, a)$ denotes action-value or Q function. $Q^\mu(s, a)$ is used in small discrete action space. For deterministic policy, the Q function can be calculated with dynamic programming as follows:

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu_{t+1})]. \quad (1)$$

Deep Q network (DQN) [10] utilizes deep learning to estimate a non-linear Q function $Q_w(s, a)$ parameterized by w. The strategy of *reply buffer* is adopted to gain independent and identical distribution of samples for training. Moreover, DQN asynchronously updates a *target network* Q_w^{tar} to minimize the least square loss as follows:

$$\begin{aligned} L(w) &= \mathbb{E}_{r_t, s_t \sim E}[(Q_w(s_t, a_t) - y^t)^2], \\ y^t &= r(s_t, a_t) + \gamma Q_w^{tar}(s_{t+1}, \mu(s_{t+1})). \end{aligned} \quad (2)$$

Policy gradient is employed to handle continuous or high dimensional actions. To estimate the parameter θ of π_θ, we maximize the expected return from the start states $\mathbb{E}_{r_i, s_i \sim \pi}[R_1] = \mathbb{E}_{r_i, s_i \sim \pi}[V^\pi(s_1)]$, which is reformulated as: $\mathbb{E}_{r_i, s_i \sim \pi}[V^\pi(s_1)] = J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a) \mathrm{d}s \mathrm{d}a$ where $V^\pi(s_1)$ is the value function of the start state. $\rho^\pi(s') = \int_{\mathcal{S}} \sum_{t=1}^{T} \gamma^{t-1} p_1(s) p(s \rightarrow s', t, \pi) \mathrm{d}s$ is the discounted state distribution, where $p_1(s_1)$ is the initial state distribution and $p(s \rightarrow s', t, \pi)$ is the probability at state s' after transition of t time steps from s. Policy gradient learns the parameter θ by the gradient $\nabla_\theta J(\pi_\theta)$ which is calculated using the *policy gradient theorem* [**?**],

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) \mathrm{d}a \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\pi, a \in \pi_\theta}[\nabla_\theta log \pi_\theta(a|s) Q^\pi(s, a)]. \end{aligned} \quad (3)$$

Actor-critic [**?**] combines the advantages of Q-learning and policy gradient to achieve accelerated and stable learning. It consists of two components: (1) an actor to optimize the policy π_θ in the direction of gradient $\nabla_\theta J(\pi_\theta)$ using Equation 3 and (2) a critic to estimate an action-value function $Q_w(s, a)$ with the parameter w through Equation 2. Finally, we obtain the policy gradient denoted as $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \rho^\pi, a \in \pi_\theta}[\nabla_\theta log \pi_\theta(a|s) Q_w(s, a)]$.

In an off-policy setting, actor-critic estimates the value function of $\pi_\theta(a|s)$ by averaging the state distribution of behavior policy $\beta(a|s)$ [**?**]. Instead of considering stochastic policy $\pi_\theta(s|a)$, deterministic policy gradient (DPG) theorem [**?**] proves that policy gradient framework can be extended to deterministic off-policy $\mu_\theta(s)$, which is given as follows:

$$\begin{aligned} \nabla_\theta J_\beta(\mu_\theta) &\approx \int_s \rho^\beta(s) \nabla_\theta \mu_\theta(s) Q^\mu(s, a) \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\beta}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a = \mu_\theta(s)}]. \end{aligned} \quad (4)$$

Deep deterministic policy gradient (DDPG) [11] adopts deep learning to learn the actor and critic in mini batches which store a replay buffer with tuples (s_t, a_t, r_t, s_{t+1}). To ensure the stability of Q values, DDPG uses a similar idea as *target network* of DQN to copy the actor and critic network: $\mu_\theta^{tar}(s)$ and $Q_w^{tar}(s, a)$. Instead of directly copying the weights, DDPG uses a "soft" target updates:

$$\theta^{tar} \leftarrow \tau\theta + (1 - \tau)\theta^{tar}, w^{tar} \leftarrow \tau w + (1 - \tau) w^{tar}, \quad \texttt{where} \quad \tau \ll 1. \quad (5)$$

III. Detailed Data Pre-processing

We use the dataset collected from an international competition on growing cucumbers in greenhouses hemming2019remote. Five teams use AI algorithms to remotely control the greenhouse to grow cucumbers during a 4-month-period. A team of experienced growers also participate in the competition as reference. This dataset contains comprehensive recordings of various quantities during the competition such as greenhouse temperature and CO_2 level.

Concretely, various aspects of the growing process are grouped into six categories and stored in six files for each team – *"CropManagement.csv"* which records management of the cucumber crop in a weekly level, *"Greenhouse_climate.csv"* which records various quantities related to the climate of the greenhouse in a 5-minute level, *"Irrigation.cvs"* which stores water usage and status during the process in a daily level, *"Production.csv"* which contains status of the cucumber crop in a daily level, *"ResourceCalculations.csv"* which records consumption of various resources in a daily level, and *"vip.csv"* which stores setpoints of controllable variables of the greenhouse. For a more detailed description of each

TABLE I
IMPORTANT NOTATIONS

Notation	Description
K	the number of variables controlled by the policies
$\mathcal{A}, a$ the amount of the control variable	caction space, $\mathcal{A} \in \mathbb{R}^K$,$a \in \mathcal{A}, a \in [0, 1]$ indicates
$\mathcal{S}, s$	cstate space, $s \in \mathcal{S}$ consists of temperature, humidity and etc. (see Table II for detail)
r and the fruit weight as our reward function.	creward function: We weighted the source cost
γ importance of immediate and future rewards	c$\gamma \in [0, 1]$ is a discount factor to balance the
$\mu_\theta(s)$	deterministic policy learned from policy gradient
$\pi_\theta(a\|s)$	stochastic policy learned from policy gradient
$\mu(s)$	a greedy policy learned from Q-learning
$\hat{\mu}(s)$	unknown policy of a doctor
$\beta(a\|s)$ generate trajectories for off-policy learning	cbehavior policy,
c $Q^\mu(s, a)$, $Q^\pi(s, a)$ after taking action a following policy μ or π	cQ-values, the expected return at state s
$Q_w(s, a)$	estimated Q function
c$V^{\mu_\theta}(s)$, $V^\pi(s)$ discounted reward from state s	cstate value, the expected total

TABLE II
BASIC STATISTICS OF STATE DEFINED IN THIS PAPER.

State	Units
GlobRad	W/m^2
outside Temp	°C
outside humidity	%RH
inside temp	°C
outside humidity	%RH
Inside CO2 concentration	ppm
Are the lamps on	0,1
Lee side vent opening	0-100%
Cumulative Irrigation	liter/m^2
Cumulative Amount of drain	liter/m^2
Action (hour-level setpoint)	**Units**
CO_2 concentration	ppm
Temp	°C
Humidity	%RH

quantity in each file, see the *"ReadMe.pdf"* file in the dataset [1].

The guideline for pre-processing those files is that we want to align all files to have the same level of granularity and each quantity represents a certain aspect of the growing process at the corresponding time moment. Following this guideline, we pre-process all the six files for each team with this 5-step recipe:

1) Organize each file to have same quantities across six teams. In particular, the "vip.csv" file for one team, "AiCU", has multiple columns of setpoints for some control variables. For each of them, we replace its corresponding columns with their average.
2) Fill in missing values (namely NaN values) caused by various unknown reasons. For those files recorded with 5-minute interval, we use backward filling. For those files recorded with daily interval, we use linear interpolation to fill in missing values.
3) Compress or extend each file to hourly level interval. For files with 5-minute interval, it means compressing the number of records by taking averages over 12 consecutive rows. For files with daily interval, it means extending the number of records. We do this by either linearly interpolating between two consecutive records with 23 extra rows of records if this quantity represents some states or status (e.g. PH of the drainage), or "diluting" each quantity by dividing it by 24 and setting it and these 23 extra rows to this result if this quantity represents some cumulative gaining (e.g. water supply).
4) Normalize each quantity in each file over all six teams. Different quantities have different units and hence different scales, so we use a linear transformation to normalize each quantity in each file over all six teams to the range of [0, 1].
5) Save all processed files.

Finally, we construct the state space, action space, and reward signal from those processed files.

- States: we merge processed "Greenhouse_climate.csv" and "Irrigation.csv" files to construct our state representation. We add one extra column "hour of day" (0, 1 through 23 and repeat) to the merged file. Since all files are aligned in time from previous treatment, a simple concatenation like this forms our state representation at each and every time step $S_t, t \in \{1, 2, \ldots, T\}$.
- Actions: we use processed "vip.csv" file as our action representation. Each row in this file forms the representation of action taken at that time step $A_t, t \in \{1, 2, \ldots, T\}$.

[1]Publicly available at www.xxx.com.

- Reward signal: we use processed "Production.csv" and "ResourceCalculations.csv" to synthesize our reward signal. We use the "Total_Prod_cum" quantity in "Production.csv" which represents the hourly growth in weight of the cucumber crop, multiply it by the price of cucumber, and get the positive reward at step time. Similarly, we multiply the each quantity in "ResourceCalculations.csv" with their corresponding price [2] and add them together to get the negative reward (resource consumption) at each time step. Finally, we multiply the negative reward by a hyperparameter C [3] and add it to the positive reward to get the overall reward at each time step $R_t, t \in \{1, 2, \ldots, T\}$.

A. Single Variable Optimization Test

Given a greenhouse cucumber planting simulator, we want to search for a parameter setting that gives optimal cucumber production. A brute-force grid search is impossible due to the large number of possible parameter settings. One special case that makes the search process easy is when all the parameter can be optimized individually. That is, parameters are independent from each other and changing one will not affect the optimal value for another. In this case, the optimal parameter setting can be quickly found by a simple greedy approach – optimize one parameter at a time and iterate over all parameters. Can our simulator use this strategy to search for an optimal parameter setting? We do not know but we hope so. Hence, we performed a validity test to our simulator.

At each time step, there is k parameters we can set and the total number of time steps (growing process time span) is T. Then one parameter setting for the simulator has $k * T$ components. In order to test whether these $k * T$ components can be individually optimized, we will select two components x_i, x_j and plot the change of final cucumber production y when these two parameters are set to different values. If the highest value for y occurs across different x_j's occurs at the same x_i, then it validates the individual optimization strategy.

We chose the temperature setpoint parameter and relative humidity setpoint parameter, and conducted three such tests. In each one of the test, we varied the temperature setpoint parameter in the range of

B. AI for agriculture

AI powered agriculture has been studied extensively since the millennium. Researchers have tried various techniques in machine learning in agriculture. Some great general review can be found in [12] and [13]. More specifically, deep learning techniques and, more recently, reinforcement learning techniques have been applied in agriculture as well.

For deep learning and agriculture, [14] provides an excellent survey to start with, and applications with deep learning includes crop classification [15], crop disease detection [16] [17] [18], remote crop count [19] [20], etc.

When it comes to reinforcement and agriculture, researchers focus more on operational management (i.e. learning a good sequence of decision making to maximize some utility). For example, [21] and studied management on wheat crop with reinforcement learning technique. Reinforcement learning also applies to resource consumption management. For instance, [22] , [23], and [24] tried to use reinforcement learning techniques optimize to maize irrigation and water system management in agriculture. A set of research work focus on design a greenhouse model and utilize dynamic programming models to learn a control policy [4]–[6]. However, these methods are all require to access to interact with the environment or a greenhouse model. Thus, these policies highly count on the greenhouse model. In addition, these dynamic programming methods can only control single variable.

Also, there are some work on agriculture with robotics as well, where mobile robots are internally powered by some AI algorithms to facilitate its jobs. In our opinion, this direction of research is promising in the sense that, conceivably, autonomous robots can significantly boost the management of large-scale crop farms as human labor becomes increasingly expensive and conventional machineries are not intelligent enough. A few work down the line are [25] and [26] where they discussed online learning of robots and vision navigation of mobile robots.

C. Reinforcement learning

The gap between the behavior policies and target polices has been verified to obstruct the performance of the traditional off-policy reinforcement learning algorithms [27], such as DQN [10] and DDPG [11]. However, In the real world. Taking exploration and exploitation to obtain trajectories for training is quite expensive. That is we can only learn a policy from a fixed size of training trajectories generated from some behavior policies.

Behavior cloning [28] which focuses on learning a policy based on a set of fixed expert trajectories, which is a form of supervised learning. It's goal is to learn a direct mapping from the states to the actions. BC can avoid interacting with the environment. However, BC is known to fail when encounter sub-optimal trajectories [7], [8], [29]. In addition, without substantial correction during training, BC is known to have compounding error [30] .

In an off-policy setting, actor-critic estimates the value function of $\pi_\theta(a|s)$ by averaging the state distribution of behavior policy $\beta(a|s)$ [**?**]. Instead of considering the stochastic policy $\pi_\theta(s|a)$, the deterministic policy gradient (DPG) theorem [**?**] proves that policy gradient framework can be extended to find deterministic off-policy $\mu_\theta(s)$, which is given as follows:

$$\begin{aligned} \nabla_\theta J_\beta(\mu_\theta) &\approx \int_s \rho^\beta(s) \nabla_\theta \mu_\theta(s) Q^{\mu\theta}(s,a) \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^\beta}[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu\theta}(s,a)|_{a=\mu_\theta(s)}]. \end{aligned} \tag{6}$$

[2]Prices come from the documentation "Autonomous control of greenhouse compartment in Bleiswijk".

[3]C controls the trade-off between maximizing cumulative reward regardless of resource consumption (C=0) and maximizing the net profit (C=1).

Deep deterministic policy gradient (DDPG) [11] adopts deep learning to learn the actor and critic in mini batches which store a replay buffer with tuples (s_t, a_t, r_t, s_{t+1}). To ensure the stability of Q values, DDPG uses a similar idea as the *target network* of DQN to copy the actor and critic networks as $\mu_\theta^{tar}(s)$ and $Q_w^{tar}(s,a)$. Instead of directly copying the weights, DDPG uses a "soft" target update:

$$\theta^{tar} \leftarrow \tau\theta + (1-\tau)\theta^{tar},\ w^{tar} \leftarrow \tau w + (1-\tau)w^{tar}, \quad (7)$$

$$\texttt{where} \quad \tau \ll 1. \quad (8)$$

IV. Methods

A. Optimize the target policy via DDPG

In this paper, we utilize DDPG to optimize the climate control policy. Specifically, A critic Q_{θ_1} is used to evaluate the different contributions of the actions given the states, which is optimized by the least square loss. The learned policy π_{θ_2} is optimized by policy gradient theory.

The objective functions of the policy π_θ is to maximize the expected Q-value which is defined as follows:

$$\mathcal{L}_{\phi_2} = -\sum_{i=1}^{M}\sum_{t=1}^{T}(Q_{\theta_1}(s_{i,t}, \pi_{\theta_2}(s_{i,t}, z)) \quad (9)$$

The objective of the Q-network is shown as follows:

$$\begin{aligned}\mathcal{L}_\beta = \frac{1}{2}\sum_i\sum_t &(Q_{\theta_1}(s_{i,t}, a_{i,t}) \\ &- (r_{i,t} + Q_\beta(s_{i,t+1}, \pi_{\theta_2}(s_{i,t+1}, z))))^2\end{aligned} \quad (10)$$

B. Optimization Details

The pseudocode is shown in algorithm 1.

Algorithm 1 RL for Climate Control

Require: behavior policy batch $\mathcal{B}$, # epochs N, batch size V, regularization weight λ_p, α_p.

1: Initialize DDPG model π_{θ_2} and Q_{θ_2} via the historical datatset;
2: **for** $epochs = 0$ to E **do**
3: Given the current state s_t of the environment, take an action a_t, $a_t = \pi_{\theta_2}(a_t|s_t)$ via the policy π_{θ_2}.
4: obtain the $(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$ tuple and add it into $\mathcal{B}$
5: $q_1 = Q_{\theta_1}(s_{i,t}, a_{i,t})$
6: $q_2 = Q_{\theta_1}(s_{i,t+1}, \pi_{\phi_2}(s_{i,t+1}, z))$.
7: Calculate the loss in Equation 9 and Equation 10 and update π_{θ_2} and Q_β.
8: **end for**

V. Experiments

A. Comparison methods

- **Five teams policies**: We extract the setpoints (actions) produced by these five teams's AI algorithms (the rank is Sonoma, iGrow, Deep_Greens, The Croperators, AiCU). Then we directly conduct these setpoints in the cucumber simulator to evaluate their performance. The ranking of these teams are
- **Behavior Cloning (BC)** Behavior Cloning (BC) is to learn a direct mapping from the states to the actions. BC can avoid interacting with the environment.
- **DDPG [11]** DDPG is a standard off-policy reinforcement learning algorithm for deterministic and continous action decision making problem. It consists of an actor and a critic, where the actor is used to produce the action and the critic is used to evaluate the actor.
- **Constrained DDPG** Constrained DDPG is the initial version of CPRL, where we only utilize the vae model with flat-decoder and without periodic regularization.

TABLE III
Model comparison on Cumulative harvest of fresh weight (CUM-FW) and Cumulative harvest of dry weight (CUM-DW).

Method	CUM-FW	CUM-DW
Sonoma	42.89	0.727
Human	35.047	1.017
iGrow	42.02	1.241
Croperators	38.39	1.119
AiCu	39.86	1.174
Deep Greens	37.19	0.902
DDPG (on policy)	44.46	1.198

B. Performance Comparison

Table III, Figure 1 and Figure 2 show the performance comparison of different methods on the cucumber simulator provided by an agricultural university. They build this simulator several years for cucumber research.

We observe that: (1) The performance of each behavior policy is different. And Sonoma achieves the best performance on Cumulative harvest (fresh weight) CUM-FW among the five teams which is coordinated with the real-world cucumber growing games. But it obtain a less weight on Cumulative harvest (dry weight) CUM-DW. Here larger dry weight indicates the cucumber contains more nutrition constituent. The reason would be that Sonama utilizes less CO_2 compared with the other methods (shown in Figure 2). (2) Human achieves less CUM-FW compared with other methods but has high CUM-DW. The reason would be that Human takes higher temp in greenhouse climate control which would accelerate the evaporation of cucumber's water. (3) DDPG is consistently outperforms the other methods on CUM-FW and CUM-DW. The reasons are (i) Due to the different quality of the behavior policy trajectories. DDPG considers reward signal to guide to learn a policy instead of directly mimicing these behavior policies (Compared to BC). (ii) DDPG achieves stable and large number of fruits in each unit during the planting process. Note that larger number of fruits is not indicates better performance, where the size of each of the fruits may become smaller. There is no monotonic relation among the CUM-FW, CUM-DW and the number of fruits.

C. Policies analysis

Figure 2 shows the actions taken by different methods under different time granularity (day-level and 10-day-level). We observe that both the Human policy and DDPG policy are

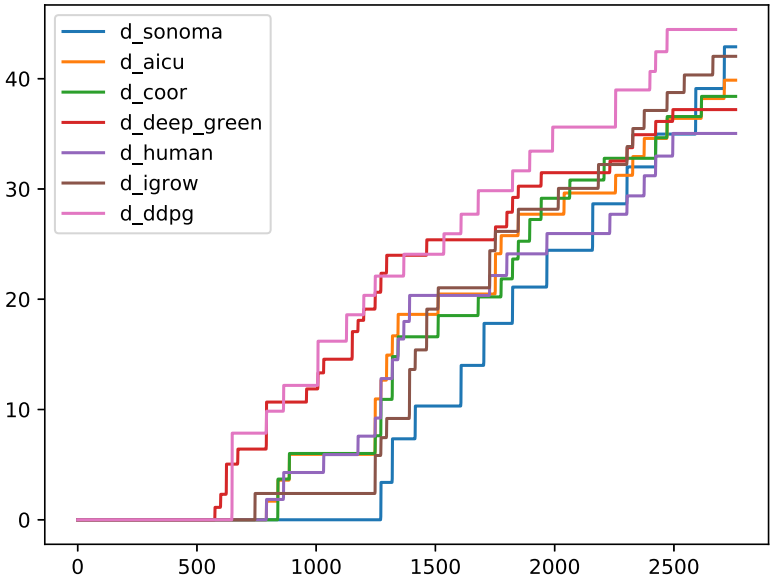

Fig. 1. Method comparison

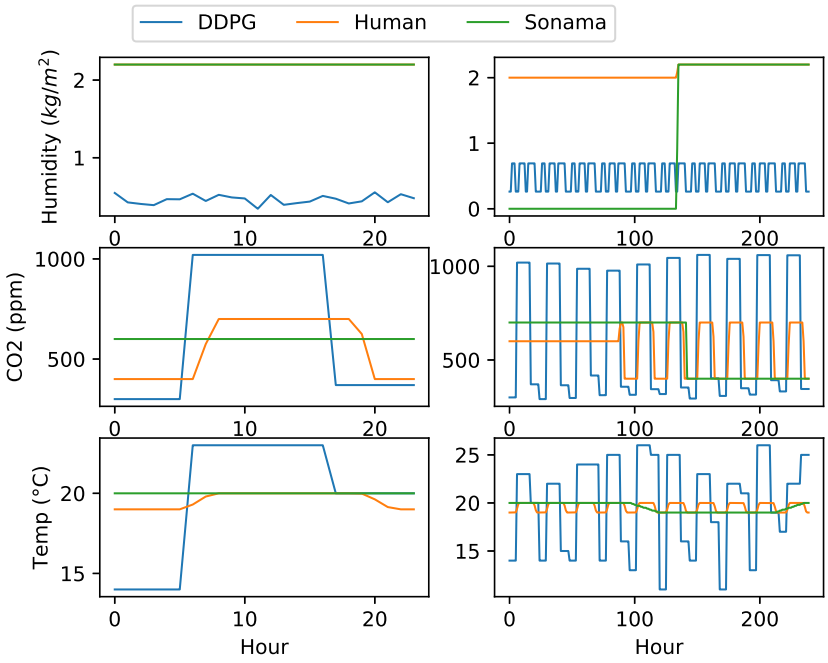

Fig. 2. Action Analysis

almost periodic. The periodic are shown in two aspects: (1) the morning, noon and night period during a day. According to Figure 2, for the morning time Human always takes the CO_2 concentration about 270 and CPRL always takes 255. (2) the period between days. For example, the Temp taken between two days are quite similar. However, the actions produced by Sonama and DDPG are not significantly periodic. hOWEVER, DDPG learns a dynamically changing policy, which would be harm for the crops to timely react to the changing environment.

VI. Conclusion

In this paper we study the problem of autonomous greenhouse control via a limited planting trajectories and without interacting with the environment. In these applications, the fundamental difficulty is to model the MDP framework. To solve this challenge, we establish DDPG method, which learn a policy in a cucumber simulator environment. We first formulate the problem of climate control as an MDP problem, followed by adopt DDPG method to learn a policy to control multiple variables. I By applying in autonomous greenhouse control problem, we demonstrate that DDPG can learn a better plant growing strategy from only 6 trajectories, while existing methods obtain much poorer results with the same settings.

However, in this study, we only control three main kinds of variables, a larger scale control variables will be considered in the future work.

VII. Acknowledgement

The work is partially supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904. We thank the anonymous reviewers for their careful reading and insightful comments on our manuscript.

References

[1] FAO, "The state of food security and nutrition in the world—building climate resilience for food security and nutrition," *Food and Agriculture Organization of the United Nations (FAO)*, 2018.

[2] R. F. . Agribusiness, "World vegetable map 2018," 2019, accessed on 11 March 2019. https://research.rabobank.com.

[3] BrainD, "What is the current state of labor in the greenhouse industry," 2018, accessed on 15 November 2018. Greenhouse Grower. Available online: https://www.greenhousegrower.com/management/what-is-the-current-state-of-labor- in-the-greenhouse-industry.

[4] Zwart, "Analyzing energy-saving options in greenhouse cultivation using a simulation model," *[S*, 1996.

[5] L. F. M. Marcelis, A. Elings, P. H. B. D. Visser, E. Heuvelink, G. Fischer, S. Magnitskiy, and S. Nicola, "Simulating growth and development of tomato crop." *Acta Horticulturae*, vol. 2009, no. 2009, pp. 101–110, 2009.

[6] A. Elings, M. Heinen, B. E. Werner, Visser, and L. F. M. Marcelis, "Feed-forward control of water and nutrient supply in greenhouse horticulture: development of a system," *Acta Horticulturae*, pp. 195–202, 2004.

[7] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *AAAI*, 2018.

[8] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots, "Fast policy learning through imitation and reinforcement," *arXiv preprint arXiv:1805.10413*, 2018.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] K. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018.

[13] J. Attonaty, M. Chatelin, F. Garcia, and S. Ndiaye, "Using extended machine learning and simulation technics to design crop management strategies," in *EFITA First European Conference for Information Technology in Agriculture, Copenhagen, DK*. Citeseer, 1997.

[14] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[15] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geoscience and Remote Sensing Letters*, 2017.

[16] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*.

[17] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[18] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, 2018.

[19] M. Rahnemoonfar and C. Sheppard, "Deep count: fruit counting based on deep simulated learning," *Sensors*, p. 905, 2017.

[20] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting apples and oranges with deep learning: A data-driven approach," *IRAL*, vol. 2, no. 2, pp. 781–788, 2017.

[21] F. Garcia, "Use of reinforcement learning and simulation to optimize wheat crop technical management," in *MODSIM*, 1999, pp. 801–806.

[22] J. Bergez, M. Eigenraam, and F. Garcia, "Comparison between dynamic programming and reinforcement learning: A case study on maize irrigation management," in *Proceedings of the 3rd European Conference on Information Technology in Agriculture*. Citeseer, 2001, pp. 343–348.

[23] B. Bhattacharya, A. Lobbrecht, and D. Solomatine, "Neural networks and reinforcement learning in control of water systems," *Journal of Water Resources Planning and Management*, pp. 458–465, 2003.

[24] A. Castelletti, G. Corani, A. Rizzolli, R. Soncinie-Sessa, and E. Weber, "Reinforcement learning in the operational management of a water system," in *IFAC workshop on modeling and control in environmental issues*, 2002, pp. 325–330.

[25] H. Hagras, M. Colley, V. Callaghan, and M. Carr-West, "Online learning and adaptation of autonomous mobile robots for sustainable agriculture," *Autonomous Robots*, pp. 37–52, 2002.

[26] J. Zhou, Q. Chen, Q. Liang *et al.*, "Vision navigation of agricultural mobile robot based on reinforcement learning." *Nongye Jixie Xuebao= Transactions of the Chinese Society for Agricultural Machinery*, vol. 45, no. 2, pp. 53–58, 2014.

[27] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *AAAI*, 2018.

[28] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, pp. 88–97, 1991.

[29] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *ICML*, 2017, pp. 3309–3318.

[30] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.