Breeding Programs Optimization with Reinforcement Learning

Omar G. Younis
ETH Zurich
omar.g.younis@gmail.com

Luca Corinzia
ETH Zurich
luca.corinzia@inf.ethz.ch

Ioannis N. Athanasiadis
Wageningen University and Research
ioannis.athanasiadis@wur.nl

Andreas Krause
ETH Zurich
krausea@ethz.ch

Joachim M. Buhmann
ETH Zurich
jbuhmann@inf.ethz.ch

Matteo Turchetta
ETH Zurich
matteo.turchetta@inf.ethz.ch

Abstract

Crop breeding is crucial in improving agricultural productivity while potentially decreasing land usage, greenhouse gas emissions, and water consumption. However, breeding programs are challenging due to long turnover times, high-dimensional decision spaces, long-term objectives, and the need to adapt to rapid climate change. This paper introduces the use of Reinforcement Learning (RL) to optimize simulated crop breeding programs. RL agents are trained to make optimal crop selection and cross-breeding decisions based on genetic information. To benchmark RL-based breeding algorithms, we introduce a suite of Gym environments. The study demonstrates the superiority of RL techniques over standard practices in terms of genetic gain when simulated in silico using real-world genomic maize data.

1 Introduction

Crop breeding programs aim to create new cultivars with desired traits by controlled mating among individuals in a population (1). Crop breeding played a vital role in the *Green Revolution* that took place in the 20th century, which resulted in a threefold increase in plant-based food production within 50 years, while only raising land usage by 30% (2). However, current growth trends do not meet the forecasted 2050 demand. Moreover, evidence shows that yields for staple crops are plateauing or degrading mainly due to climate change and soil degradation (3).

The decrease in genotyping cost has produced abundant crop genomic data, unveiling novel opportunities to adapt crop genetics for these adverse conditions. Recent advancements have focused on predicting phenotype traits from genomic data for quicker and cheaper crop selection without expensive and slow trials (4; 5; 6). However, relying solely on estimated traits can lead to low-diversity pools and compromise long-term breeding program success.

Data-driven methods for decision-making can fill this gap and allow breeding programs to retain genetic diversity and adapt to long-term objectives, e.g., climate adaptation. To use these methods to optimize crop breeding, we can frame a breeding program as a sequential decision-making problem where, at each breeding generation, the breeder takes some actions, e.g., which and how many plants to select, and how to cross them. Crossing two plants produces offspring with a mixture of genes from the two parents. The process is repeated for several generations to deploy a cultivar with desired traits.

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2023.

From this perspective, we can optimize design choices of breeding programs with Reinforcement Learning (RL) (7), a paradigm to address sequential decision-making problems.

Contributions: In this paper, we develop a framework to optimize the design choices of breeding programs, with three main contributions. (1) In Section 2, we model a breeding program as a Markov Decision Process (MDP), allowing us to leverage techniques from the RL literature to adaptively optimize the design choices. (2) For this scope, we present in Section 4 a package containing multiple RL environments that simulate breeding programs with different sets of actions and observations. (3) In Section 5, we show the effectiveness of the approach in selecting plants based on a learned score; this approach outperforms the standard genomic selection based on the estimated traits. Please note that the results presented are based on simulations and may differ in real-world applications.

Related work: Machine learning models have been applied in breeding programs in various ways (8), however, often for trait prediction in genomic assisted breeding (9; 10). To the best of our knowledge, RL has been only used in the context of breeding in (11), where the authors optimize the population size at every step (generation) of the breeding program, while other decisions (e.g., genomic prediction function) are considered fixed.

2 Problem definition

In reinforcement learning, sequential decision-making problems are formalized as Markov decision processes (MDPs) (12), i.e. a tuple $(S, \rho, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where: S is the set of all possible states; ρ is the probability distribution of the initial state s_0; $\mathcal{A} : S \to A$ maps each state s to the possible actions in s; $\mathcal{P}$ denotes the transition probability $\mathbb{P}(s_{t+1} \mid s_t, a_t)$ with $s_t, s_{t+1} \in S, a_t \in \mathcal{A}(s_t)$; $\mathcal{R} : S \times A \to \mathbb{R}$ is the reward function that defines the desirability of the state-action pair; γ (between 0 and 1) balances the preference for immediate versus long-term rewards.
A policy π is a function that maps a state s_t to a distribution over valid actions in the set $\mathcal{A}(s_t)$. RL algorithms aim to find a policy that maximizes the expected return, i.e.,

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}_{\mathcal{P},\pi,\rho} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right], \tag{1}$$

where T is the length of the episode, and Π is the set of possible policies. In the case of breeding programs, at each generation t, the breeder performs arbitrary crosses obtaining a new population. We define the state at time step t as the set of genomes of the population

$$s_t = \{g_1^{(t)}, g_2^{(t)}, ..., g_{n_t}^{(t)}\} \in S \tag{2}$$

where g_i^t represent the genome of the i-th individual at generation t, and n_t is the population size at generation t. We explain in the next section how we represent a genome g_i^t in silico. Let's denote the set of indices of the population at step t as $[n_t] = \{1, ..., n_t\}$. An action at timestep t consists of making an arbitrary number of crosses from pairs of plants from the set $[n_t]$, i.e.

$$\mathcal{A}(s_t) = \{(e_1, ..., e_l) \mid l \geq 1, e_i \in [n_t] \times [n_t]\}\,. \tag{3}$$

Given such action, the population at the subsequent timestep has size l, hence $n_{t+1} = l$. The transition probability $\mathbb{P}(s_{t+1}|s_t, a_t)$ describes the biological process of the cross (genetic recombination) according to action a_t. Since a single breeding cycle can take several years and RL requires many interactions with the environment, we simulate the transition probability *in silico*.

3 Genetic simulation

Our experiments focus on crops with two full sets of chromosomes, called diploid crops (the setting is easily extensible to other ploidy). Among all the nucleotides in a chromosome, we are interested in those varying among the same type of crop, called Single Nucleotide Polymorphisms (SNPs). In particular, we are interested in the SNPs affecting the trait of interest. We assume every SNP has two possible configurations, hence they can be represented by a boolean array of size $(m, 2)$, where m is the total number of SNPs and the second axis is due to diploidy. When two plants are mated, the offspring inherits one chromosome set from each parent. The inherited chromosome is formed by a stochastic biological process called genetic recombination, which involves the mixing of genetic

material from the parental chromosome. Simulating this process involves processing large arrays and can be slow. We rely on CHROMAX (13), a performance-oriented breeding simulator based on JAX (14), which allows us to easily parallelize the operations on GPU(s).

The estimated trait value (e.g. yield) is computed using a linear regression model trained on real data:

$$\hat{y} = \sum_{i=1}^{m} w_i \cdot (g_{i,1} + g_{i,2}) \tag{4}$$

where m is the total number of SNPs, w_i is the linear regressor weight for the SNPs i, and $g_{i,1}, g_{i,2}$ are the boolean values of the i-th SNPs for the first and second chromosome, respectively.

4 Gym environments

We introduce a set of Gym environments (15) to train RL agents to learn breeding program design choices at different levels of complexity. Utilizing the vectorization capabilities of JAX (14), the environments parallelize and potentially distribute computations across hardware accelerators.

BREEDINGGYM: The base environment, where the agent observes the genomes of the population and can perform all possible crosses. The observation space is a variable-sized sequence of arrays. The action size is also dynamic, represented as a sequence of integer pairs as defined in Equation (3). During initialization, the user can specify the genetic data, the horizon T, and whether the agent should be rewarded during the episode or solely at the conclusion. Rewards are computed using aggregation functions (e.g., max, mean) of trait values over the population, estimated from the genetic data.

SIMPLIFIEDBREEDINGGYM: This environment fixes the size of the population, resulting in static dimensions of observations and actions. The observation is a dictionary containing, for each plant, the yield and the genome correlation coefficient. The action is a dictionary containing the number of plants to select based on a given trait and how many random crosses to perform on them. To keep the number of plants constant at each generation, the crosses can produce more than one offspring.

SELECTIONSCORES: Genomic selection in breeding often focuses on trait values when selecting plants (e.g., higher yields). Yet, this might ignore the potential of certain plants with poor current performance. Selection scores like optimal haploid value (16) are based on this intuition. This environment is aimed at learning a non-myopic selection score. The observation is the genome, while the action is an array containing a score for each plant. The k top-scoring plants are chosen and crossed randomly. The user fixes the population size, the value k, and the number of crosses l.

PAIRSCORE: In this environment, the agent assigns a score for every possible cross. This allows for more degrees of freedom in choosing which parents have more potential if mated together. Thus, for a population of size n, the action will be a matrix of size $n \times n$. Then, n crosses are selected according to the score matrix, where n is fixed by the user.

5 Experiments

We present here the results obtained on the SELECTIONVALUES environment with fixed population size $n_t = 200$. At every step, we select $k = 20$ plants based on the scores in the action and we make 10 random crosses with 20 offspring each. Additionally, we fix the number of generations to $T = 10$, and we reward the agent based only on the estimated trait of the best individual of the last generation.

Data and observations: We use maize genetic data from (17) which contains the SNPs values of a population along with the linear regressor weights w, to predict the Shoot Apical Meristem (SAM) volume (18), which we use as a proxy for yield. In every episode, we sample 200 plants to compose the initial population. To reduce memory usage and speed up the simulation, we randomly subset a total of $m = 1000$ SNPs. Consequently, the observation is an array with shape $(200, 1000, 2)$. To ease training, we process the observation by multiplying it with w. In this way, the estimated SAM volumes can be obtained by summing the values over the last two axes of the observation array.

Algorithm: We use the Proximal Policy Optimization (19) algorithm implemented in StableBaselines3 (20). To make the policy invariant to the permutation of plants in the population, we use a neural network that processes the plants independently and outputs the score, i.e. a scalar value.

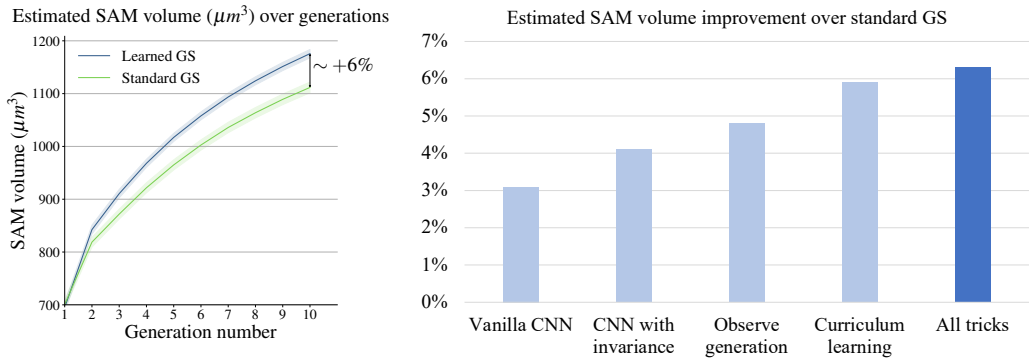

Figure 1: *(On the left)* Estimated SAM volume in μm^3 during the breeding program with standard and learned GS, averaged across 100 trials, with shaded area indicating standard error.
(On the right) Estimated SAM volume percentage increase by applying the tricks explained in this section. Note: on "Curriculum learning" we also let the agent observe the generation number.

Architecture: To keep SNPs adjacency information while reducing the number of policy parameters, we process the observation with a 1-dimensional convolutional neural network. We also include information about the generation number as the percentage of completion of the episode. More information about the architecture can be found in Appendix A.

Training and results: To help the agent handle sparse reward, we do curriculum learning (21) by gradually extending the time horizon from $T = 3$ to $T = 10$ during training. We trained for 7 million generations on a *NVIDIA GeForce RTX 2080 Ti*, taking around 24 hours. Figure 1 shows the increase in the estimated SAM volume during the 10 generations of the breeding program and the effectiveness of the employed tricks. Our genomic selection method (Learned GS) obtains an estimated $1176 \mu m^3$ SAM volume, around 6% higher than the one obtained by standard genomic selection (Standard GS).

6 Discussion

This paper introduces a new approach to optimize breeding programs. We formulate the problem as a Markov decision process, making techniques from the RL literature applicable to this challenge. We show the effectiveness of the method by training an RL agent to select plants during the breeding program. To foster research in this field, we publish a collection of Gym environments that allow researchers to optimize various aspects of breeding programs. The genetic processes are simulated, enabling swift and cheap exploration of potential solutions. However, it is important to acknowledge the gap between simulated processes and reality, which can lead to considerable performance degradation.

Beyond the societal relevance of breeding scheme optimization, these environments present a set of interesting challenges for RL research. Specifically, the observation space can be very large as well as the action space, which scales combinatorially with the population size. Moreover, their size is in general dynamic and can be decided by the agent; in this case, we typically need to introduce constraints that reflect real-world costs and budgets. Moreover, the MDP is factored (22) as the state is composed of weakly connected components (the genomes of the plants). Finally, the reward can be non-stationary as the regressor model is continuously trained with new data.

We hope this work can foster collaborations across artificial intelligence and the agriculture community, and drive further advancements in the optimization of breeding programs. Our efforts seek to make a tangible impact on agricultural practices and contribute to addressing global challenges related to food security and sustainability.

References

[1] Covarrubias-Pazaran, G., Z. Gebeyehu, D. Gemenet, et al. Breeding schemes: What are they, how to formalize them, and how to improve them? *Frontiers in Plant Science*, 12, 2022.

[2] Pingali, P. L. Green revolution: Impacts, limits, and the path ahead. *Proceedings of the National Academy of Sciences*, 109(31):12302–12308, 2012.

[3] Ray, D. K., N. Ramankutty, N. D. Mueller, et al. Recent patterns of crop yield growth and stagnation. *Nat Commun*, 3:1293, 2012.

[4] Meuwissen, T. H., B. J. Hayes, M. E. Goddard. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*, 157(4):1819–1829, 2001.

[5] Crossa, J., P. Pérez-Rodríguez, J. Cuevas, et al. Genomic selection in plant breeding: methods, models, and perspectives. *Trends in plant science*, 22(11):961–975, 2017.

[6] Dreisigacker, S., J. Crossa, P. Pérez-Rodríguez, et al. Implementation of genomic selection in the cimmyt global wheat program, findings from the past 10 years. *Crop Breeding, Genetics and Genomics*, 3(2):e210004, 2021. Doi: 10.20900/cbgg20210005.

[7] Sutton, R. S., A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[8] Yoosefzadeh Najafabadi, M., M. Hesami, M. Eskandari. Machine learning-assisted approaches in modernized plant breeding programs. *Genes*, 14(4), 2023.

[9] Yu, T., W. Zhang, J. Han, et al. An ensemble learning approach for predicting phenotypes from genotypes. In *2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*, pages 382–389. 2021.

[10] Zhao, W., X. Lai, D. Liu, et al. Applications of support vector machine in genomic prediction in pig and maize populations. *Frontiers in Genetics*, 11, 2020.

[11] Moeinizade, S., G. Hu, L. Wang. A reinforcement learning approach to resource allocation in genomic selection. *Intelligent Systems with Applications*, 14:200076, 2022.

[12] Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[13] Younis, O. G., M. Turchetta, D. Ariza Suarez, et al. ChromaX: a fast and scalable breeding program simulator, 2023.

[14] Bradbury, J., R. Frostig, P. Hawkins, et al. JAX: composable transformations of Python+NumPy programs, 2018.

[15] Brockman, G., V. Cheung, L. Pettersson, et al. Openai gym, 2016.

[16] Daetwyler, H. D., M. J. Hayden, G. C. Spangenberg, et al. Selection on optimal haploid value increases genetic gain and preserves more genetic diversity relative to genomic selection. *Genetics*, 200(4):1341–1348, 2015.

[17] Moeinizade, S., G. Hu, L. Wang, et al. Optimizing selection and mating in genomic selection with a Look-Ahead approach: An operations research framework. *G3 (Bethesda)*, 9(7):2123–2133, 2019.

[18] Ha, C. M., J. H. Jun, J. C. Fletcher. Shoot apical meristem form and function. *Curr Top Dev Biol*, 91:103–140, 2010.

[19] Schulman, J., F. Wolski, P. Dhariwal, et al. Proximal policy optimization algorithms, 2017.

[20] Raffin, A., A. Hill, A. Gleave, et al. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[21] Narvekar, S., B. Peng, M. Leonetti, et al. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

[22] Boutilier, C., R. Dearden, M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1104–1111. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

A Architecture details

In the following, we explain the policy architecture that we used in the experiments presented in Section 5. We represent the policy network in Figure 2.

The network processes each plant in an episode independently and it comprises two components: a feature extractor for processing the genetic array data, yielding 64 features, and another network responsible for generating the scalar value, which relies on these features and the generation number. As we use Proximal Policy Optimization, we additionally utilize a value network that shares the feature extractor with the action network.

The first convolutional layer has 64 kernels of length 32 and stride 8, and the second one with 16 kernels of length 8 and stride 2. With the input having shape $(1000, 2)$ and these parameters, the flattened output has a size of 928. This output is then processed by a Multi-Layer Perceptron layer that maps the 928 array to 64 features. As there is no inherent semantic order between the two chromosomes we ensure the network's invariance to the permutation of the two channels. To achieve this, we average the output of the feature extractor considering both possible orders.

Additionally, we incorporate information about the generation number to generalize better when changing the time-horizon. We do so by creating a mapping from the completion percentage of the breeding program to 16 features. These 16 features are concatenated with the previously obtained 64 features. We use two different mapping for the action and the value networks.

The final step involves processing these combined 80 features with two MLPs (one for the action and one for the value) with 32 hidden layers and scalar outputs. Finally, to determine the value of the current state, we average the outputs (one for each plant) produced by the value head.

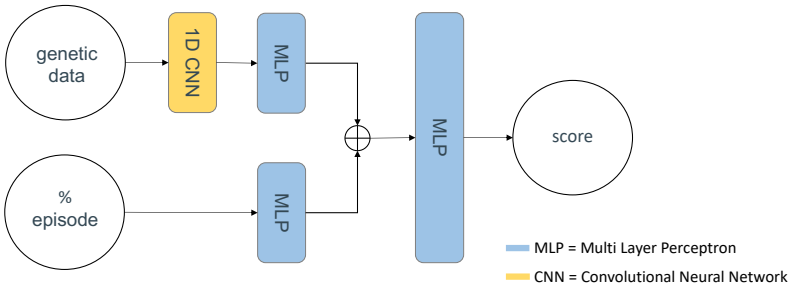

Figure 2: Representation of the policy net. The final score is a scalar value that is used by the environment to select the plants to cross.