Plant breeding optimization through Reinforcement Learning

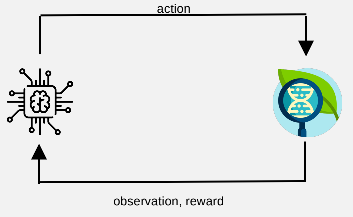
Omar G. Younis[1], Luca Corinzia[1], Matteo Turchetta[1], Steven Yates[2], Bruno Studer[2], Ioannis Athanasiadis[3]
[1]Dept. Computer Science, ETH Zurich; [2]Dept. Environmental System Science, ETH Zurich; [3]WDCC, WU

1 Introduction

Plant breeding programs allowed increasing in food production over the past years. However, we need to double the annual yield improvement to match the future demand due to population growth. Even worse, there is concern that yield improvement is beginning to plateau[1]. We aim to apply Reinforcement Learning to optimize breeding schemes in a data-driven manner.

2 Method Overview

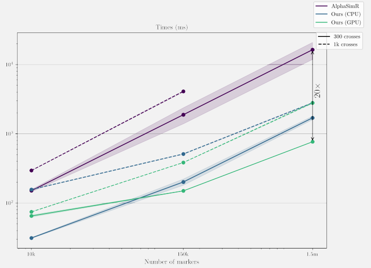
- Reinforcement Learning is data hungry and requires many trials to train; we need to work with an efficient breeding simulator. We wrote our custom crossing simulator in Python based on the method described by Han et al.[2]
- Reinforcement Learning optimizes a sequence of actions while observing the state of the environment in order to maximize an objective.
- We make the Reinforcement Learning environment based on the simulator. Actions, observations, and the objective can be decided based on use cases and to iteratively increase the complexity of the task.



3 Breeding Simulator

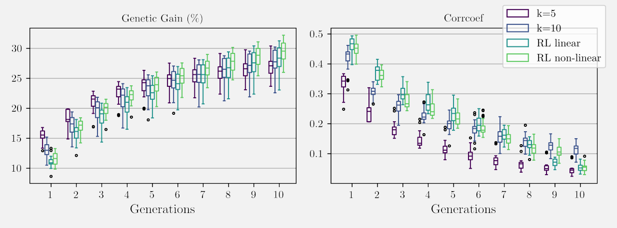
The simulator is based on JAX[3]; it runs transparently on CPU, GPU, or TPU and is automatically parallelized. Running it on a distributed system is also straightforward. We compare it with AlphaSimR[4].

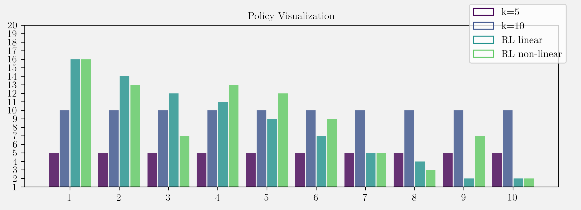
```
>>> simulator = BreedingSimulator('genetic_map.csv')
>>> population = simulator.load_population('genome.txt')
>>> simulator.GEBV(population).mean()
Yield  1945.53894
dtype: float32

>>> selected_ind = simulator.select(population, k=10)
>>> simulator.GEBV(selected_ind).mean()
Yield   2881.909668
dtype: float32

>>> new_pop = simulator.diallel(selected_ind, n_offspring=10)
>>> len(new_pop)
450

>>> cross_indices = np.array([
      [1, 5], [3, 10], [100, 2], [7, 93], [28, 41]
    ])
>>> population = simulator.cross(new_pop[cross_indices])
>>> len(population)
5

>>> dh_pop = simulator.double_haploid(population)
>>> simulator.save_population(dh_pop, "dh_genome.txt")
```



4 Results and Discussion

We trained two Reinforcement Learning agents on the most basic environment, in which only the generation number is observed, the budget is constant, and the agent chooses the number of bests to select.





5 Conclusion

- Tackling food shortage will be a fundamental challenge of the future.
- Reinforcement Learning shows promising results for improving breeding programs.
- Our simulator enables parallel computation for crossing, that is needed for training complex Reinforcement Learning agents.
- We aim to train agents with more complex actions and adaptive to the current generation's state. We want to define a use case including practical constraints (e.g. total budget or cost of actions) and publish a competition.

References

1. K. P. Voss-Fels, A. Stahl, and L. T. Hickey, "Q&A: modern crop breeding for future food security," BMC Biol., vol. 17, no. 1, p. 18, Feb. 2019, doi: 10.1186/s12915-019-0638-4.
2. Han, Y., J. N. Cameron, L. Wang, and W. D. Beavis, 2017 The predicted cross value for genetic introgression of multiple alleles. Genetics 205: 1409–1423. https://doi.org/10.1534/genetics.116.197095
3. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018)
4. Gaynor, R. Chris, Gregor Gorjanc, and John M. Hickey. 2021. AlphaSimR: an R package for breeding program simulations. G3 Gene|Genomes|Genetics 11(2):jkaa017. https://doi.org/10.1093/g3journal/jkaa017.

Partner/Sponsor: Agroscope