Final Report: Implementation and Enhancement of ARAL Software Tool

Manuel Schneckenreither

2023-12-18

1 Report Period: Year 2023

1.1 Introduction:

During the year 2023, significant progress has been made in the implementation, enhancement and evaluation of the Open Source Software tool ARAL, as well as its' deep-learning successor library LARA. LARA is designed to conduct Reinforcement Learning (RL) based dynamic program analysis within the context of AUTOSARD. In particular substantial process has been made for the following four points:

1. Implementation of the Open Source Software Tools ARAL and LARA. ARAL is a tabular-based reinforcement learning package that was built for the development, validation and evaluation of the same-titled average reward adjusted reinforcement learning algorithms. For preliminary testing purposes preliminary function approximators, like artificial neural networks, have been integrated in the ARAL library.

2. Validation of the solution quality, including a sensitivity analysis, for tabular-based algorithms implemented in the ARAL.

3. Implementation of the LARA library, which is a performant implementation of a high-level machine learning library including features like processing on CUDA-enabled GPU devices, data conversion functionalities, easily configurable supervised learning algorithms, like Feedforward ANNs and (Variational) Auto-Encoders [3], as well as reinforcement learning algorithms with corresponding additions, like replay memories. A major part of the work on LARA was the enhancement and development of a corresponding average reward adjusted deep reinforcement learning algorithm ARPPO, that lifts the novel ARAL algorithm to deep learning. Hence, LARA is designed to conduct RL based dynamic program analysis. Furthermore, the library includes other state-of-the-art reinforcement algorithms, one of them being Proximal Policy Optimization (PPO) [5], whose ideas were fundamental for the development of ARPPO.

4. Testing and Validation of ARPPO in comparison to other deep RL algorithms, like Deep RL and PPO.

1.2 Implementation of ARAL and Test Cases

Building on a previously developed implementation of ARAL, we cleaned the code and improved its performance. We also added the average reward R-learning algorithm for comparison purposes. Additionally, we added following new example environments suitable for a tabular representation of the state-action value function: (i) the CartPole environment, (ii) the discrete-action mountaincar environment, (iii) a demand forecasting environment including seasonal demands, (iv) a newsvender inventory environment known from operations management and applied economics, and (v) a one-way-membrane example. After preliminary tests, we have chosen the CartPole and the MountainCar environments as the ones that most effectively show the advantages of ARAL over the other RL algorithms.

1.3 Testing Solution Quality of ARAL

Extensive testing was conducted to assess the solution quality of the tabular-based ARAL algorithm, including a sensitivity analysis. Different instances of the RL algorithm were applied to evaluate its performance under various conditions. Evaluation reports were generated to document the outcomes of these tests. In particular, we have thoroughly evaluated the performance of the algorithms Q-Learning, R-Learning and ARAL, for which following reports were generated (with 30 replications and statistical analysis per report):

1. Printer-Mail example
    - Evaluation until divergence, due to low complexity of the problem
2. Gridworld example (without random reward and with random reward)
    - Sum Reward and Average Steps to Goal
    - Sensitivity Analysis: random reward
    - Sensitivity Analysis: initial learning rate
    - Sensitivity Analysis: discount factor γ_1
3. Admission Control Queuing System
    - Sum Reward and Queue Length
    - Sensitivity Analysis: discount factor γ_0
4. CartPole Balancing
    - Sum Reward and Average Steps per Episode
5. MountainCar
    - Reward and Average Steps to Goal

1.4 Machine Learning Library LARA to Integrate deep RL Models

An important milestone achieved during this period was the expansion of the base functionality to seamlessly integrate deep RL models. In order to ensure compatibility and a generalizable design we opted for a complete re-implementation of the library, where we call these re-implementation “LARA” (library for average reward algorithms). The new design allows a flexible implementation for different kind of data structures and is designed with the concept of re-usable code, whenever possible.

The developemnt of the LARA involved utilizing the Hasktorch library, a pytorch pendant for Haskell. The new library, LARA, includes features like data conversion functionalities, easily configurable supervised learning algorithms, like Feedforward ANNs and (Variational) Auto-Encoders [3], as well as reinforcement learning algorithms with corresponding additions, like replay memories. In recent years variational auto-encoders have been increasingly used to learn a model of the enviornment, as e.g. proposed by the DreamerV3 algorithm [2]. One of the most time-extensive works was to enhance and develop of a corresponding average reward adjusted deep reinforcement learning algorithm ARPPO, which is a novel deep learning implementation of the ARAL algorithm. The library also implements other state-of-the-art RL algorithms, like Proximal Policy Optimization (PPO) [5], Soft Actor-Critic (SAC) [1], and deep Q-Learning [4].

1.5 Evaluation of deep RL Models

In order to evaluate these algorithms in the de-facto standard environments, we have developed a library for interoperability with the library *gymnasium*[1].

Following the integration of deep RL models, rigorous testing and evaluation were conducted to validate their effectiveness. The performance of algorithms like PPO, SAC, and deep ARAL were assessed across different scenarios, providing valuable insights into their applicability within the AUTOSARD framework.

ARAL’s functionalities were thoroughly tested and validated in the context of Work Packages WP A(4), B(3), and WP (3,4). This involved ensuring that ARAL met the specific requirements outlined in these work packages and demonstrated compatibility with the overall project objectives.

1.6 Conclusion

In conclusion, the outlined activities have significantly contributed to the advancement of ARAL/LARA as a powerful tool for RL in the context of dynamic program analysis. The successful integration of deep RL models enhances the tool’s capabilities, opening new possibilities for advanced program analysis.

[1]`https://github.com/Farama-Foundation/Gymnasium`

Possible next steps can be further refinement of ARAL and incorporating feedback from testing. Additionally, ongoing collaboration with the relevant work packages will be maintained to ensure alignment with project objectives.

References

[1] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[2] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.