Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)
Demonstrations Track

RLOP: A Framework for Reinforcement Learning, Optimization and Planning Algorithms

Song Zhang
EII Inc., Tokyo, Japan
zhangsong0618@gmail.com

Abstract

Reinforcement learning, optimization, and planning/search are interconnected domains in artificial intelligence. Algorithms within these domains share many similarities. They complement each other in solving complex decision-making problems, and also offer opportunities for cross-disciplinary integration. However, conducting research on algorithms across these domains typically requires learning the specialized libraries. These libraries often couple algorithms with domain-specific problem classes, making it difficult to conduct cross-disciplinary researches. In order to solve this problem, we developed a generic and lightweight framework for reinforcement learning, optimization, and planning/search algorithms (RLOP). It implements only the core logic of algorithms, abstracting away domain-specific details by defining interface functions, which enables flexible customization and efficient integration across different domains. The framework has been open-sourced at https://github.com/songzhg/RLOP.

1 Introduction

In artificial intelligence, reinforcement learning (RL), optimization (Opt), and planning/search have relatively independent research background and application scope. Reinforcement learning studies how an agent ought to take actions in a dynamic environment in order to maximize the cumulative reward. Optimization spans a wider range of fields including artificial intelligence and operations research. Its problem is to optimize an optimization function subject to certain constraints [Ryan, 2003]. Planning can be divided into state-space planning and plan-space planning. State-space planning is viewed primarily as a search through the state space for an optimal policy or an optimal path to a goal. Plan-space planning is instead a search through the space of plans [Sutton and Barto, 2018].

There are similarities and overlaps between reinforcement learning, optimization, and planning/search. For example, reinforcement learning and planning/search are all based on looking ahead to future events, computing backed up values, and then using it to update an approximate value function [Sutton and Barto, 2018]. Similarly, optimization algorithms such as the simplex and the branch and bound methods can also be considered as planning/search methods within the solution space. These algorithms can complement each other in solving complex decision-making problems. Specifically, optimization methods are efficient in solving problems with well-defined objectives and constraints. Planning extends this by considering sequences of actions over time. Reinforcement learning is well-suited for complex environments with long-term planning. In recent years, several cross-disciplinary studies have found success, including AlphaZero [Silver *et al.*, 2017], which combines the Monte Carlo Tree Search (MCTS) with reinforcement learning to master board games, and the use of reinforcement learning or MCTS to solve combinatorial optimization problems [Bello *et al.*, 2016] [Zhang *et al.*, 2022]. The integration between these domains is a promising direction for future researches.

Currently a generic framework of reinforcement learning, optimization, and planning/search is not very common. Conducting research on algorithms across these domains typically requires learning the specialized algorithm libraries separately. These libraries often implement algorithms with domain-specific problem classes. For example, Stable Baselines3 [Raffin *et al.*, 2019], a reinforcement learning library, uses a well-defined environment class as the input of algorithms. It simplifies the applications of algorithms, but makes customization across domains difficult, as the environment class dose not efficiently fit the algorithms from other domains. To solve this problem, we developed a generic and lightweight framework for reinforcement learning, optimization, and planning/search algorithms (RLOP). It implements only the core logic of algorithms, abstracting away domain-specific details by defining interface functions, rather than directly accessing information from problem-specific classes. This design enables flexible customization and efficient integration across different domains.

The rest of the paper is organized as follows. In section 2, we introduce the design principle of RLOP. In section 3, we compare RLOP with other related libraries. We show some examples of using RLOP to solve benchmark problems in section 4, and conclude in section 5.

```
class MCTS {
    // The number of child states
    virtual int NumChildStates() const = 0;

    // Check if a node is fully expanded
    virtual bool IsExpanded(const Node&) const = 0;

    // Revert the current state
    virtual void RevertState() = 0;

    // Make a step on the current state
    virtual bool Step(Int child_i) = 0;

    // Get the reward of the current state
    virtual double Reward() = 0;
};
```

Figure 1: Examples of interface functions of MCTS. Users implement these functions based on the problem to solve.

2 Framework Design

RLOP is a generic and lightweight framework for reinforcement learning, optimization, and planning/search algorithms, aimed at simplifying studying, comparing and integrating algorithms across domains. It is developed in C++ to ensure fast or memory-efficient implementation of algorithms. And it also includes implementations of benchmark problems, such as Snake game, Connect4 [Dabas *et al.*, 2022], vehicle routing problem (VRP) and multi-armed bandit.

RLOP uses libtorch [Paszke *et al.*, 2019] for constructing deep learning models. And it provides a C++ interface for gymnasium [Brockman *et al.*, 2016] by using pybind11 [Jakob *et al.*, 2017], making it possible to run reinforcement learning algorithms on python-based gym environments.

2.1 Features

There are two major features of the design:

- Simple: RLOP implements only the core logic of the algorithm, with other improvements being added in an extensible manner. Simple choices are preferred to more complex ones.
- Domain-independent: RLOP abstracts away domain-specific details of algorithms by defining interface functions. For each new problem, users are required to implement these functions manually. This design decouples algorithms from domain-specific problem class (such as the "Env" class in reinforcement learning). Figure 1 shows the example of interface functions of Monte Carlo Tree Search (MCTS).

2.2 Implemented Algorithms

The algorithms currently implemented in RLOP are listed as follows (also in Table 1):

1. Deep Q Network (DQN) [Mnih *et al.*, 2013]: an off-policy reinforcement learning algorithm making use of replay buffer and target network to stabilize the learning process.
2. Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017]: an on-policy reinforcement learning algorithm combining ideas from A2C (having multiple workers) and TRPO (using a trust region for actor).
3. Soft Actor Critic (SAC) [Haarnoja *et al.*, 2018]: an off-policy reinforcement learning algorithm maximizing a trade-off between expected return and entropy of policy with a stochastic actor.
4. Monte Carlo Tree Search (MCTS): a search algorithm using random sampling for deterministic problems. PUCT is a variant of MCTS taking prior probability into consideration. The root-parallel version of MCTS and PUCT are also implemented.
5. Alpha-beta Search (Alpha-beta) [Knuth and Moore, 1975]: a search algorithm for a game tree pruning away branches that cannot possibly affect the final decision. Transposition table (TT) is used to record positions that have already been explored to avoid repeat exploration.
6. Tabu Search [Glover, 1990]: a meta-heuristic search used to solve optimization problems, which can escape the local optimum by banning visited solutions or movements.
7. Simulated Annealing (SA) [Bertsimas and Tsitsiklis, 1993]: a probability based meta-heuristic for approximating the global optimum of a given function.

Algorithm	Type
DQN	RL
PPO	RL
SAC	RL
MCTS/PUCT	Search
Root-parallel MCTS/PUCT	Search
Alpha-beta	Search
Alpha-beta with TT	Search
Tabu Search	Opt
SA	Opt

Table 1: Algorithm List

3 Related Work

In this section, we introduce two widely recognized libraries related to reinforcement learning, optimization, and planning/search. The algorithms implemented in these libraries are listed in Table 2.

Stable Baselines3 (SB3) is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It encapsulates complex implementations internally, allowing beginners to experiment without being buried in implementation details, but makes hacking into the algorithms and conducting cross-domain research more difficult.

OpenSpiel [Lanctot *et al.*, 2019] is a collection of environments and algorithms for research in general reinforcement learning and search/planning in games. It offers a rich set of algorithm implementations including reinforcement learning, multi-agent reinforcement learning (MARL), tabular-based methods, optimization and planning/search. Unfortunately, the library lacks support for domains other than games.

Algorithm	RLOP	SB3	OpenSpiel
A2C		○	○
DQN	○	○	○
PPO	○	○	○
SAC	○	○	
DDPG		○	
HER		○	
TD3		○	
MCTS/PUCT	○		○
Alpha-beta	○		○
Alpha-beta with TT	○		
Tabu Search	○		
SA	○		
Tabular methods			○
MARL methods			○
Opt in games			○

Table 2: Comparison between libraries (○: included)

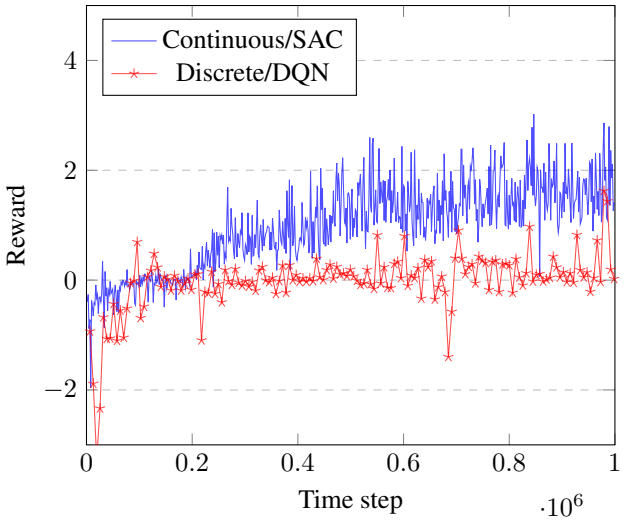

Figure 2: The changes of the average reward of a batch during training in lunar lander (Continuous/Discrete).

Algorithm	Train Time (s)	Train Step	Eval (Reward) Mean	Eval (Reward) Std
SAC	191.22	10^6	**282.73**	**22.14**
DQN	**44.58**	10^6	152.48	72.84

Table 3: Comparison between SAC and DQN

4 Case Study

In this section, we show examples of using RLOP to solve benchmark problems and illustrate the results. Because of the page limit, we only investigate two problems provided in RLOP, including lunar lander and vehicle routing problem (VRP). For more examples, please refer to https://github.com/songzhg/RLOP. All the experiments are run on the platform with Intel Core i9-12900K CPU and Nvidia GeForce RTX 3090 Ti GPU.

4.1 Lunar Lander

Lunar lander is a classic rocket trajectory optimization problem, which is part of the Box2D environments of gymnasium. The algorithm is required to control the engines of a rocket on three directions to make a smooth landing. It includes both discrete and continuous versions. We implement a DQN agent for the discrete version and a SAC agent for the continuous version based on RLOP. The changes of the average reward of a batch during training are presented in Figure 2. We evaluate each algorithm on 1000 episodes. The results are shown in Table 3. We also conduct informal comparative experiments between RLOP and Stable Baselines3. Roughly, even though RLOP uses pybind11 to indirectly invoke python-based gym environments, it can still achieve at least 1.4 times the efficiency of Stable Baselines3.

4.2 Vehicle Routing Problem

Vehicle routing problem (VRP) is a classic combinatorial optimization problem, which involves finding routes with the lowest cost for a fleet of vehicles to a set of customers. We implement insertion (a heuristic based method), local search, simulated annealing and tabu search based on RLOP to solve two sets of problems named as C50 (10 vehicles, 50 customers) and C200 (10 vehicles, 200 customers) respectively. Each set contains 100 randomly generated problems, with each vehicle having a different depot. We first run the insertion algorithm to generate a initial solution and then use other algorithms to improve it. The average cost and computing time are presented in Table 4. As is shown, the tabu search and simulated annealing get relatively better results, but with longer computing time.

5 Conclusion

In this paper, we introduced a generic, lightweight and open-sourced framework for reinforcement learning, optimization, and planning/search algorithms (RLOP), aimed at simplifying studying, comparing and integrating algorithms across domains. It implements only the core logic of algorithms, abstracting away domain-specific details by defining interface functions, which enables flexible customization and efficient integration across different domains. In this paper, we presented the algorithms implemented in RLOP and demonstrate their applications in solving benchmark problems. As an open-sourced project, we anticipate further enhancements and updates to RLOP in the future.

Algorithm	C50 Cost	C50 Time (ms)	C200 Cost	C200 Time (ms)
Insertion	509.02	**0**	1077.43	**0.01**
Local Search	423.99	5.86	901.32	173.28
SA	419.25	34.34	**837.96**	461.12
Tabu Search	**368.23**	18.47	899.7	705.01

Table 4: Comparison between algorithms on VRP

References

[Bello *et al.*, 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Bertsimas and Tsitsiklis, 1993] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Dabas *et al.*, 2022] Mayank Dabas, Nishthavan Dahiya, and Pratish Pushparaj. Solving connect 4 using artificial intelligence. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 1*, pages 727–735. Springer, 2022.

[Glover, 1990] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[Jakob *et al.*, 2017] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11–seamless operability between c++ 11 and python. *URL: https://github.com/pybind/pybind11*, 2017.

[Knuth and Moore, 1975] Donald E Knuth and Ronald W Moore. An analysis of alpha-beta pruning. *Artificial intelligence*, 6(4):293–326, 1975.

[Lanctot *et al.*, 2019] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[Raffin *et al.*, 2019] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[Ryan, 2003] Conor Ryan. Computer algorithms. In Robert A. Meyers, editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 507–523. Academic Press, New York, third edition edition, 2003.

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

[Silver *et al.*, 2017] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Zhang *et al.*, 2022] Tianyu Zhang, Amin Banitalebi-Dehkordi, and Yong Zhang. Deep reinforcement learning for exact combinatorial optimization: Learning to branch. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3105–3111. IEEE, 2022.