ORSuite: Benchmarking Suite for Sequential Operations Models

Christopher Archer Siddhartha Banerjee Mayleen Cortez
Carrie Rucker Sean R. Sinclair[*] Max Solberg
Qiaomin Xie Christina Lee Yu

ABSTRACT

Reinforcement learning (RL) has received widespread attention across multiple communities, but the experiments have focused primarily on large-scale game playing and robotics tasks. In this paper we introduce *ORSuite*, an open-source library containing environments, algorithms, and instrumentation for operational problems. Our package is designed to motivate researchers in the reinforcement learning community to develop and evaluate algorithms on operational tasks, and to consider the true *multi-objective* nature of these problems by considering metrics beyond cumulative reward. [1]

1. INTRODUCTION

Reinforcement learning (RL) is a natural model for problems involving real-time sequential decision making, including inventory control, resource allocation, ridesharing systems, and ambulance routing [16, 2, 25, 18, 28]. In these models, an agent interacts with a system that has stochastic transitions and rewards, and aims to control the system by maximizing their cumulative rewards across the trajectory. Reinforcement learning has been shown in practice to be an effective technique for learning complex control policies [20].

These sequential decision making problems have been considered across multiple communities: machine learning, computer science, statistics, economics, and operations research. These fields are converging recently, primarily because as data becomes more readily available and computing power improves, it allows for customizing domain-specific algorithms. As a result, the new zeitgeist for the field is developing *data-driven decision algorithms*: algorithms which adapt to the structure of information, constraints, and objectives in any given domain. This paradigm highlights the importance of taking advantage of domain knowledge to help design algorithms that scale to real-world problem instances.

In addition, many of the problems arising in operations research are naturally multi-objective. Any algorithm should compete on the **trifecta for RL in operations**: large average rewards with mild storage and computational complexity. Each model itself also has additional metrics which algorithms should compete for, whether that be minimizing the mean response time in an ambulance dispatch system or the efficacy in a resource allocation problem. Moreover, many of the problems naturally have continuous or combinatorial state and action spaces, which makes designing RL algorithms for these domains an important direction for the research community.

[*]Contact author at `srs429@cornell.edu`.
[1]The project is available at `https://github.com/seanrsinclair/ORSuite`

1.1 Our Contributions

In this note we outline the structure and purpose of the *ORSuite* package [21]. Our goal is to provide a standardized library for the reinforcement learning community to explore applying algorithms to problems arising in operations research. The package is aimed at modeling common operational problems, providing implementation of existing heuristic approaches to these problems, and optional instrumentation to benchmark the performance of algorithms across a variety of metrics. To this end, we include three main components:

OpenAI Gym Environments for OR Models: We provide robust OpenAI Gym [7] environment implementations for several problems arising in operations research, including ambulance routing, resource allocation, vaccine allotment, and ridesharing systems (as outlined in Section 2).

OR-Based Heuristic Algorithms: For each of the included environments, we complement them by providing implementation of existing heuristic algorithms to better benchmark the performance of reinforcement learning algorithms. We additionally include implementation for discretization based algorithms for settings when the state-action space are continuous [23, 22, 26].

Optional Instrumentation: We have optional instrumentation for running simulations in the episodic finite horizon setting. Our instrumentation collects trajectory information for calculating domain-specific metrics, along with average rewards, storage, and computational complexity. In addition, we provide automatically generated plots which provide a comparison between algorithms on each of the metrics. We also include a wrapper to run experiments with the *stable baselines* package, which provides implementation for neural-network based reinforcement learning algorithms [10].

Our package complements existing environment implementations [11, 17] by including additional instrumentation and metrics beyond average or cumulative rewards. The *ORSuite* package is designed for developing and testing RL algorithms applied to operational problems, and we will discuss a case study in Section 3. We welcome any feedback and collaborations as we continue to iterate and develop the package.

2. ENVIRONMENTS INCLUDED

Due to space considerations we discuss four environments currently implemented in the package: ambulance routing, resource allocation, vaccine allotment, and ridesharing system models. However, this is an incomplete list as we are also working on incorporating inventory control with lead times and multiple suppliers, queuing networks, and airline revenue management. Our environments serve as an addition to those developed in [11, 17], focusing more on stochastic problems and scenarios which are not currently solved by existing theory in operations research. Our models extract out the essential features from the problem to highlight operational considerations, and in principal it is easy to modify to make them more realistic [19].

2.1 Ambulance Routing

This is as a stochastic variant of the popular grid-world or metrical task-system problem. A fleet of $k \in \mathbb{N}$ ambulances try to dynamically re-position themselves in a region to minimize travel time to arriving service requests. The algorithm receives ambulance positions from the environment, and chooses locations to station each ambulance, paying a cost for relocation. Next a request is drawn from some underlying distribution, after which the closest ambulance travels to the demand location [14, 8, 5].

We provide two underlying state-action spaces: the unit interval $X = [0, 1]^k$ (each ambulance can be anywhere on the unit interval); and a graph environment where the user defines an undirected weighted graph $G = (V, E)$, and the space is $X = V^k$ (each ambulance can be at any node). The goal of the agent is to minimize the weighted costs of traveling to the action chosen and responding to a call. In addition to allowing users to specify the arrival distribution and the graph for ambulance requests, we also include a graph and dataset representing requests from the Ithaca, New York community.

Additional Metrics: The default rewards are calculated via a convex combination of the cost to travel and service a request and the cost to re-station the ambulance. We also include the mean response time and the variance in mean response time to the requests as additional metrics.

Heuristic Algorithms: For the line environment we include two heuristic algorithms.

Static Agent: This algorithm never moves the ambulances at the beginning of the iteration.

Median Agent: This uses the collected dataset of past arrivals sorted by the arrival location, partitions it into k quantiles, and selects the middle data point in each quantile as the location to station the ambulances.

For the graph environment we include *Static Agent* and *Median-Like Agent*: Stations ambulances at the nodes that would have minimized the distance traveled to respond to all calls that have arrived in the past.

Mode Agent: Stations ambulances at the nodes where calls have most frequently occurred in the past.

2.2 Fair Resource Allocation

This environment is motivated by a problem faced by a collaborating food-bank (Food Bank for the Southern Tier of New York (FBST) [9]) in operating their mobile food pantry program [24]. In these systems, the mobile food-bank must decide on how much food to allocate to a distribution center on arrival, and without knowledge of demands in future locations. This model also extends to broader stockpile allocation problems (such as vaccine and medical supply allocation) and reservation mechanisms.

We consider a principal tasked with fairly dividing K resources among T rounds. Each person arriving has a type and utility function, characterizing how satisfied they are with respect to the allocation given to them. Over each round $t \in [T]$, the number of individuals of each type is drawn from a (known) distribution and observed by the principal. The principal then decides an allocation for individuals of each type before proceeding to the next round. Allocation decisions are irreversible, and must obey the overall budget constraints. The goal of the agent is to design an online allocation strategy which minimizes various fairness metrics.

Additional Metrics: In allocation problems, the offline solution which optimizes the Nash Social Welfare (defined as the product of utilities for individuals of each type for the allocation received) can be shown to satisfy three Varian-fairness criteria: pareto-efficiency (for any individual to benefit, another must be hurt), proportionality (each individual prefers their own allocation to an equal allocation), and envy-freeness (no individual prefers another's allocation to their own) [29, 30, 27]. In the online setting, due to uncertainty in demand, achieving an exactly fair allocation is impossible. The default rewards for the environment are chosen to be the per-round Nash Social Welfare. We also include additional measures of hindsight proportionality and envy-freeness as outlined in [24], excess resources, and the ex-post distance to the optimal fair allocation in hindsight.

Heuristic Algorithms: We include three algorithms [24]:

Equal Allocation: uses the equal allocation solution replacing unknown quantities with their expectation.

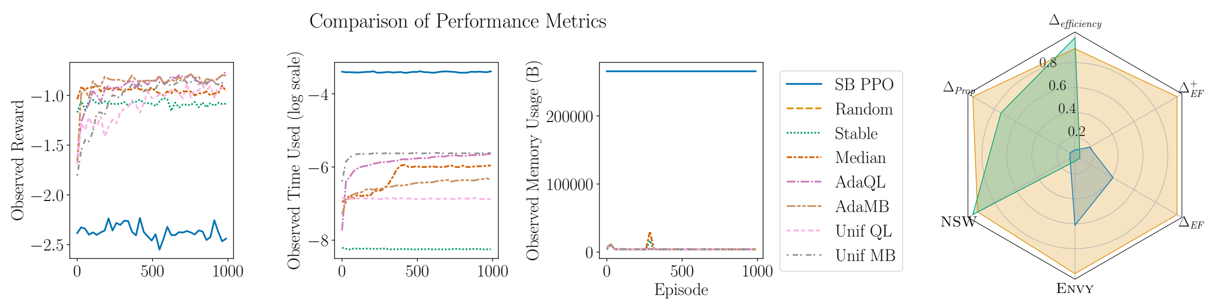
Fixed Threshold: uses concentration to generate a lower bound on the optimal solution in hindsight and allocates according to that at every round.

Hope-Guardrail: uses both a lower and upper bound on the optimal allocation in hindsight and greedily allocates according to the upper bound to minimize waste while simultaneously ensuring enough resources are saved to allocate at least the lower solution to everyone in the future.

2.3 Vaccine Allocation

This environment seeks to understand the design of optimal allocation policies of vaccines to a population modeled with the SIR epidemic model [1, 12, 13] for the spread of disease. We look at a population of size K partitioned into four risk classes: medical workers, non-medical essential workers, high-risk individuals and low-risk individuals. Each risk class is further divided into two groups: susceptible and infected but asymptomatic. Additionally, we consider vaccinated people to be part of a recovered group.

We consider the scenario where an agent chooses a priority order of the four risk classes to vaccinate the population. For example, if the priority order is $\{3, 2, 1, 4\}$, vaccines are administered to susceptible individuals in risk class 3 until there are no vaccines left or there are no more susceptible people in risk class 3. If the latter happens, vaccines are administered to susceptible individuals in risk class 2, then 1 and finally to 4. This directly mimics the phased approach seen in the COVID-19 vaccine roll-out in New York [15]. The goal for this environment is to decide a vaccination allocation strategy which minimizes the time until the epidemic is



(a) Ambulance Routing (b) Resource Allocation

Figure 1: Left: Sample line plots for the ambulance routing environment. Right: Sample radar plots with the additional metrics for the resource allocation problem. Legends are omitted due to space constraints as the plots serve to highlight the instrumentation provided.

eliminated.

Heuristic Algorithms: We include a fixed-priority algorithm which administers vaccines according to a fixed priority order across the whole time horizon.

Additional Metrics: The metric included in the environment measures the number of new infections that occurred during the transition. As such, the metric calculates the total number of infections that occur throughout the horizon.

2.4 Ridesharing Systems

This environment studies the design of assignment controls in networks with a fixed number of circulating resources. Each time a demand arises, the algorithm chooses a supply node which can service the request. If the demand is served, the supply unit then relocates to the "destination" of the demand. We frame this as a model of ridesharing systems, but the environment can be used more generally for closed-loop control [4, 3, 6].

This environment can be thought of as a discrete version of the rideshare dispatch problem with instantaneous travel times in which we have supply units (cars) spread across a graph that attempt to satisfy demand units (ride requests) that arrive at each time step. We work with a bipartite compatibility graph $G = (V_S \cup V_D, E)$ such that K supply units (cars) are distributed over the supply nodes V_S and demand units (ride requests) arrive at the demand nodes V_D. At every timestep, a request (i, j) is observed from a node in V_D to V_S. The algorithm then decides a valid supply node to service the demand request.

Heuristic Algorithms: We will implement *Scaled Max Weight* policies which dynamically manage the distribution of supply in the network. These policies are parameterized by a vector of scaling factors for each supply node, and demand is serviced by assigning a supply from a compatible node with the largest scaled supply units. The policies are simple and have been shown to perform well in realistic simulations.

Additional Metrics: We incorporate the ability for a user-defined reward function. Typical models of rideshare dispatch systems focus on minimizing unmet rides. However, other possible options would be to incorporate the distance traveled to meet the demands (incorporating wait times that individuals experience while waiting for their request to be fulfilled).

3. OPTIONAL INSTRUMENTATION

Included in *ORSuite* is optional instrumentation aimed at providing performance metrics for an algorithm when applied to an environment. We include four main features:

Episodic Finite Horizon Instrumentation: Runs simulations in the finite horizon setting between an arbitrary agent and environment, collecting trajectory information and performance metrics.

Wrapper for Stable Baseline: Runs simulations in the finite horizon setting between agents from the *stable baseline* package and an environment, collecting trajectory information and performance metrics [10].

Plot Generation: Generates figures (similar to Fig. 1) comparing the performance of different algorithms on a specific environment on the basis of chosen metrics.

Teaching Interface: A command-line interface with visualizations where *you are the agent*, which can be used as a teaching tool for describing the reinforcement learning problem set-up.

These tools are designed for developing and testing new RL algorithms applied to our operations research environments as highlighted in Section 2. The wrappers for *stable baselines* allows for testing and developing new neural-network based RL algorithms for these settings, and the instrumentation and automatically-generated plots provide performance benchmarks across a variety of metrics.

As an example, in Fig. 1 on the left hand side we ran a sample simulation on the ambulance routing problem on the line as outlined in Section 2.1. We compared the performance of the heursistic algorithms described in Section 2.1 with discretization-based algorithms from [23, 22, 26] and the proximal policy optimization algorithm from [10]. These plots help to highlight the true multi-objective nature of many of these operations research problems. While neural network algorithms can often be shown to achieve great performance in reinforcement learning, they require hyper-parameter tuning and increased costs with respect to time complexity and space complexity. Similarly in Fig. 1 on the right we ran a sample simulation on the resource allocation problem as outlined in Section 2.2. These radar plots help highlight the need for incorporating domain-specific knowledge when developing an algorithm to achieve good guarantees with respect to a specific metric.

4. REFERENCES

[1] Daron Acemoglu, Victor Chernozhukov, Iván Werning, and Michael D Whinston. Optimal targeted lockdowns in a multi-group sir model. Working Paper 27102, National Bureau of Economic Research, May 2020.

[2] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. 2020.

[3] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. *Operations Research*, 2021.

[4] Siddhartha Banerjee, Yash Kanoria, and Pengyu Qian. Dynamic assignment control of a closed queueing network under complete resource pooling. *arXiv e-prints*, pages arXiv–1803, 2018.

[5] Allan Borodin, Nathan Linial, and Michael E Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 39(4):745–763, 1992.

[6] Anton Braverman, Jim G Dai, Xin Liu, and Lei Ying. Empty-car routing in ridesharing systems. *Operations Research*, 2019.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[8] Luce Brotcorne, Gilbert Laporte, and Frederic Semet. Ambulance location and relocation models. *European journal of operational research*, 147(3):451–463, 2003.

[9] Food Bank of the Southern Tier of New York. `https://www.foodbankst.org/`, 2020.

[10] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[11] Christian D Hubbs, Hector D Perez, Owais Sarwar, Nikolaos V Sahinidis, Ignacio E Grossmann, and John M Wassick. Or-gym: A reinforcement learning library for operations research problem. *arXiv preprint arXiv:2008.06319*, 2020.

[12] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London*, 115:772, 1927. Series A, Containing Papers of a Mathematical and Physical Character.

[13] Anup Malani, Satej Soman, Sam Asher, Paul Novosad, Clement Imbert, Vaidehi Tandel, Anish Agarwal, Abdullah Alomar, Arnab Sarker, Devavrat Shah, Dennis Shen, Jonathan Gruber, Stuti Sachdeva, David Kaiser, and Luis M.A. Bettencourt. Adaptive control of covid-19 outbreaks in india: Local, gradual, and trigger-based exit paths from lockdown. Working Paper 27532, National Bureau of Economic Research, July 2020.

[14] Matthew S Maxwell, Mateo Restrepo, Shane G Henderson, and Huseyin Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.

[15] New York State Department of Health. New york state's covid-19 vaccination program. October 2020.

[16] W Powell. Reinforcement learning and stochastic optimization, 2019.

[17] Antoine Prouvost, Justin Dumouchelle, Maxime Gasse, Didier Chételat, and Andrea Lodi. Ecole: A library for learning inside milp solvers. *arXiv preprint arXiv:2104.02828*, 2021.

[18] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[19] Samuel Ridler. Jemss. `https://github.com/uoa-ems-research/JEMSS.jl`, 2021.

[20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[21] Sean Sinclair, Christopher Archer, Carrie Rucker, Max Solberg, Mayleen Cortez, Shashank Pathak, Siddhartha Banerjee, and Christina Yu. Orsuite. `https://github.com/seanrsinclair/ORSuite`, 2021.

[22] Sean Sinclair, Tianyu Wang, Gauri Jain, Siddhartha Banerjee, and Christina Yu. Adaptive discretization for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[23] Sean R. Sinclair, Siddhartha Banerjee, and Christina Lee Yu. Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–44, Dec 2019.

[24] Sean R Sinclair, Gauri Jain, Siddhartha Banerjee, and Christina Lee Yu. Sequential fair allocation of limited resources under stochastic demands. *arXiv preprint arXiv:2011.14382*, 2020.

[25] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

[26] Zhao Song and Wen Sun. Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*, 2019.

[27] Robert Sugden. Is fairness good? a critique of varian's theory of fairness. *Nous*, pages 505–511, 1984.

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

[29] Hal R. Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9(1):63–91, September 1974.

[30] Hal R Varian. Two problems in the theory of fairness. *Journal of Public Economics*, 5(3-4):249–260, 1976.