Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 221 (2023) 526–532

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Conference on Information Technology and Quantitative Management

Multi-objective Reinforcement Learning – Concept, Approaches and Applications

Linzi Zhang[a,b,c,d*], Zhiquan Qi [a,c,d], Yong Shi [a,c,d]

[a]School of Economics and Management, University of Chinese Academy of Sciences, Beijing, 100190, China
[b] University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom
[c]Research Center on Fictitious Economy & Data Science, Chinese Academy of Sciences, Beijing 100190, China
[d]Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing 100190, China

Abstract

Real-world decision-making tasks are generally complicated and require trade-offs between multiple, even conflicting, objectives. As the advent and great development of advanced information technology, it has evolved into using reinforcement learning (RL) algorithms to tackle the multi-objective decision making (MODM) problems. In this paper, we will first identify the basic concepts and factors when modelling the MODM tasks with reinforcement learning, and then review the traditional RL, such as Sarsa, Q-Learning, Policy Gradients, Actor-Critic, Monte-Carlo learning, and modern deep RL algorithms applied in this process. Furthermore, the specific practical scenarios described in MODM problems will be summarized through analyzing some typical articles. Finally, the future trends of multi-objective reinforcement learning will be discussed.

Keywords: Multi-objective decision making; Reinforcement Learning; Survey

1. Introduction

Multi-objective decision making, also named as multi-criteria decision making or multi-criteria optimization, has long been a significant topic in most real-world decision problems, since we inherently care about more than one aspect when evaluating and optimizing the strategic decisions [1]. For instance, whether a major technological project can be implemented should consider its economic benefits, social benefits, production safety, environmental protection and other objectives, leading to a multi-objective decision making (MODM) problem. Researchers and decision-makers have been dedicated to developing methods and techniques to balance the trade-off among these criteria and explore the optimal strategies. With these multiple objectives, most algorithms focus on combining all the important

10.1016/j.procs.2023.08.018

criteria together into a single, scalar, additive objective function, such as Weighted-sum method and Natural Evolution Strategies (NES) [2]. However, this semi-blind manual mechanism would damage the interpretability of the decision-making process and cannot handle the always changing preferences. As an insight to address these issues, the Pareto optimal frontiers are widely applied to provide a portfolio of any possible optimal solutions [3].

Research on planning approaches to retrieve Pareto frontiers of MODM has been established for a long time. Multi-objective particle swarm optimization (MOPSO) algorithm, Analytic Hierarchy Process (AHP) models, and the Convex Hull Value Iteration (CHVI) algorithms have been developed to search for the Pareto optimal solution sets in the feasible solution regions [4]. To nonetheless deal with the multiple objectives of the real world, as identified by Wiering and De Jong [5], those approaches have issues of computational feasibility and non-stationarity, while a common design is to regard this model as a multi-objective Markov decision process (MOMDP) and find the Pareto solutions sequentially. Further, the Reinforcement Learning (RL) is introduced to solve the MOMDP model by reasoning about sets of possibly optimal value vectors and policies. The classical RL algorithms, such as Q-learning and Temporal Difference (TD) Learning, are demonstrated effective in Pareto optimal solution sets searching [6].

Recently, deep reinforcement learning (DRL) has emerged and gradually improved as a powerful technique for sequential optimization issues, which applied neural networks as the key components to capture the state transition instead [7]. It has achieved great success in focusing on relevant information area and reducing the overhead computation of deep neural network. In this paper, we will mainly review the reinforcement learning and deep reinforcement learning techniques used in multi-objective decision making, to summarize its basic concepts, specific process, and applications by analyzing typical articles. The reminder of this paper is arranged as follows: Section 2 will describe the basic concept of MODP and RL. Section 3 will summarize some typical MODP cases using RL or DRL algorithms. Section 4 mainly discusses the existing and possible innovative applications of MODP and RL.

2. Problem Setting

In this section, we will formulate the classical multi-objective optimization problem and its sequential configuration, i.e., multi-objective Markov decision process (MOMDP). Then, the basic elements of MOMDP are presented when it is modelled into reinforcement learning.

2.1. Multi-Objective Decision making

Multi-objective decision making (MODM) model always contains multiple contradictory objective functions, and it is hoped to find a solution set that can balance all optimization objectives well. Pareto optimal frontier is an ideal state of multi-objective optimization problem, which is impossible to improve the situation of some objectives without hurting any other else. The formalized definition of MODM is described in Eq. (1).

$$\begin{aligned} &min/max && f_m(x), m = 1,2,\dots,M \\ &subject\ to && g_j(x) \geq 0, j = 1,2,\dots,J \\ & && h_k(x) = 0, k = 1,2,\dots,K \\ & && x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1,2,\dots,n \end{aligned} \tag{1}$$

2.2. Multi-Objective Markov Decision Process

Further, since the Pareto frontier is hard to obtain directly, taking an explicitly sequential approach to planning and learning becomes essential to acquire optimal solutions in decision problems. Therefore, we formalize the multi-objective decision making (MODM) as a multi-objective Markov decision process (MOMDP). A MOMDP is represented by the tuple $(S, A, P, \boldsymbol{R}, \gamma)$, where:

- S: state space or state set;
- A: action set;
- P: state transition probability matrix;
- $R \in \mathbb{R}^m$: (immediate) reward function of m objective functions;
- γ: discount factor, used to compute cumulative rewards.

Specifically, the MOMDP basic framework includes an agent and the environment with which it interacts. Agents make decisions (decide what actions to take) based on the current state of the environment; This action causes a transition in the state of the environment, and the environment gives the agent a (immediate) reward based on the action and the state change it causes, presenting the agent with a new state.

2.3. Reinforcement Learning

In MOMDPs, the agent behaves according to the policy function π, where π is a $S \times A \rightarrow [0,1]$ mapping, i.e., for each state, the action is selected from a certain probability distribution. Generally, the state and action sets in many real-world problems are continuous and infinite, or too large to enumerate even if discrete, which makes the problem considerably harder. Reinforcement Learning (RL) thus applied to estimate the values of each considered policies. The value function of a policy π in a MOMDP is defined as:

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a) \tag{2}$$

where

$$Q_\pi(s_t, a) = \mathbb{E}_A[U_t | S_t = s_t, A_t = a_t] \tag{3}$$

and

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots \tag{4}$$

That is, U_t is the discounted return, which is the cumulative discounted future reward. $Q_\pi(s_t, a)$ denotes the action-value function for policy π at state s_t when taking the action a, and $V_\pi(s_t)$ is the state-value function of policy π at state s_t.

The goal of RL is typically to learn the optimal policy function. In single objective RL problems, a unique optimal value V^* exists where the corresponding optimal policies π^* with this value can be multiple. The goal in single-objective RL is typically to learn an optimal policy. While in multi-objective case, however, each objective could achieve an optimal state V^*, and there exists multiple possibly optimal value vectors $\boldsymbol{V}$. We therefore think about the Pareto dominance solution sets to MORL problems, as defined in Eq. (5).

$$Pareto(\Pi) = \{\pi \in \Pi \mid \nexists \pi' \in \Pi : V_{\pi'} \succ_p V_\pi\} \tag{5}$$

where $\succ_p$ is the pareto dominant:

$$V_{\pi'} \succ_p V_\pi \Leftrightarrow (\forall i : V_\pi^i \geq V_{\pi'}^i) \wedge (\exists i : V_\pi^i > V_{\pi'}^i) \tag{6}$$

In other words, the Pareto solution sets is the portfolio of dominant policies, where exists no other policy with value that is equal or better in all objectives.

3. Main Techniques

In this section, we will list and review several representative reinforcement learning techniques used in multi-objective decision making, including basic RL algorithms and DRL algorithms. Some classical papers are reviewed as well to illustrate the specific procedures of the approaches and its characteristics.

3.1. Basic Reinforcement Learning (RL) Algorithms

Research on planning RL algorithms to MODPs has long been established for finding Pareto set policies, such as Sarsa, Q-Learning, Policy Gradients, Actor-Critic, Monte-Carlo learning. These approaches have successfully-launched in learning the optimal policy value function.

3.1.1 Sarsa Algorithm

The State-Action-Reward-State'-Action' (Sarsa) algorithm is a kind of temporal difference reinforcement learning approach to learn the value function Q of a specific action under a specific state. Specifically, during the iteration, the agent first chooses an action A at state S referring to ε-greedy method and then transit to a new state S', where an immediate reward R is obtained meanwhile. Further, the agent will continue to choose an action A' at state S', and the value function Q is accordingly updated as shown in Eq. (7).

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A)) \tag{7}$$

where γ is the attenuation factor, and α denotes the iteration step size. In this way, the value function Q is established and optimized according to the rewards obtained by interacting with the environment [8, 9].

3.1.2 Q-Learning Algorithm

The fundamental framework of Q Learning is similar to that of Sarsa, which also enables the system to conduct exploration under the guidance of policy function, and update the status value at each step of iteration. However, Q Learning algorithm has different update formulas as listed in Eq. (8).

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_{A'} Q(S',A') - Q(S,A)] \quad (8)$$

It's distinguishable that Q-learning is uncertain about the next state and action when updating the target Q value, thus it will select the action with the largest Q value as the next step update. Therefore, the two algorithms perform totally different. Sarsa is more prudent, since it will remember every wrong exploration, and it will be more sensitive to mistakes, while Q-learning only cares about the maximization of Q value, so Q-learning will be very greedy and bold [10].

3.1.3 Policy Gradients Algorithm

The above-mentioned value-based algorithms, such as Sarsa and Q-Learning, require one more operation to calculate a maximum value, which is hard to implement in continuous or higher-dimensional spaces. This leads to the advent of policy-based methods. These value-free methods learn the possibility distribution of policy directly, and they not only avoid policy degradation due to value function errors, but also apply more easily to continuous action space problems.

Policy-based reinforcement learning is applicable with random policies and continuous action space. It starts from a fixed or random initial state, and allows the agent to explore the environment, generating a state-action-reward sequence from the starting state to the ending state, i.e., $S_1, A_1, R_1, \ldots, S_t, A_t, R_t, \ldots$. For each time stamp t, the policy distribution function exists when the maximum reward is obtained [11].

3.1.4 Actor-Critic Algorithm

Policy-based approaches also have some common disadvantages, such as low data efficiency or sample utilization, large variance and difficult to converge. To address these issues, Actor-Critic Algorithm integrates the value-based and policy-based framework, where Actor is used to predict the probability of action, Critic is to predict the reward in this state.

Combined with the method of Policy Gradient (Actor) and Vaule Function Approximation (Critic), Actor is based on probabilistic selection action, Critic (Q-learning or other value-based methods) can be used to estimate the Value of each state. Subtract the value of this state from the value of the next state, (TD-error), Critic will tell the actor that the next action will be increased and updated. If TD-error is positive, the next action will be increased and updated. Otherwise, the update range of the actor will be reduced. Critic evaluates the value of action based on the action of Actor, and the Actor modifies the probability of selecting action according to the value of Critic. To sum up, Actor is the strategy function $\pi(a \mid s)$, which is learning a policy to get as high a return as possible, while Critic refers to value function $V_\pi(s)$ to evaluate the value function of the current policy. With the help of the value function, the actor-critic algorithm can update parameters in a single step without waiting until the end of the round.

3.1.5 Monte-Carlo learning Algorithm

The policy gradient algorithm needs to acquire the Expectation of the value function, which is complicated to integrate the high-dimensional action space. Therefore, the Monte Carlo method is applied to approximate the policy gradient. Formally, at each state t, the agent chooses action A according to the policy network, and then evaluates this action based on the value function. Further, we find the Monte Carlo approximation of policy gradient and use gradient ascent to update the network.

Monte Carlo method can be understood as when the algorithm completes an iteration, then it uses the results of this round to learn and make an update. Since we already have the data for the entire iteration, and therefore the reward for each step, we can easily calculate the total future reward for each step. Thus, at each state S, its decay reward is calculated and the state value is updated finally.

3.2. Deep Reinforcement Learning (DRL) Algorithms

Deep learning has been widely used in both academic research and practical applications, which is popular in the field of computer vision [12], natural language processing [13], and voice recognition [14]. Deep Reinforcement Learning (DRL) is the combination of deep learning and reinforcement learning. Specifically, it combines the structure of deep learning and the thought of reinforcement learning to solve the decision-making problems. With the assist of powerful characterization ability of neural network, DRL performs well in fitting value or policy functions of complicated state-action space.

Taking *Super Mario* game as an example, any different position of agent and brick can be equivalent to a different state. Such a large number of states makes it impossible for traditional reinforcement learning to assign an action to each state, while deep learning can automatically extract features through end-to-end learning ability. Training a complex multi-layer model with strong expressive power to fit the current state, reinforcement learning to learn how to perform the corresponding actions according to the current state, in order to obtain the maximum cumulative rewards and punishments.

Deep Q-Network (DQN) is one of the most representative approaches of DRL. It is actually the variants of *Q Learning*, and combines it with neural network. On the whole, DQN and *Q learning* have very similar objective value and update ways of value. The main difference is that DQN combines *Q learning* with deep learning and uses deep network to approximate action value function, while *Q learning* uses table storage. DQN adopts the training method of experiential playback, randomly sampling from historical data, while *Q learning* directly adopts the data of the next state for learning [15].

Since convolutional neural networks have natural advantages in image processing, in image processing research domain, DQN uses 4 frames adjacent as the original image input, and outputs value function Q through deep convolutional neural network and fully connected neural network, realizing end-to-end learning control. While for time series information, DQN adopts recurrent neural networks to add experiential replay mechanism, capturing long- and short-term dependencies.

4. Applications

This section presents examples of complicated decision-making situations where multi-objective reinforcement learning approaches play a role. These examples are inspired by some of the aspects discussed in above sections.
Multi-objective analysis is a fundamental tool for decision makers to properly handle the possible trade-offs among a number of competing objectives. Specifically, **water reservoir operations** need to evaluate multiple conflicting objectives related to significant socio-economic criteria. On the one hand, the water supply needs to cater the livelihood of downstream residents and agricultural demands, leading to a storage in winter and release in irrigation season. On the other, residents on the shores argue to keep the horizon within a certain range for recreational services. Therefore, the regulation issue becomes complicated with the presence of multiple objectives that accordance with the two above: river navigability, hydropower production, agricultural irrigation, sightseeing, flood mitigation, and many others [16, 17].

Furthermore, considering a journey to a given destination, the decision on the means of transportation involves several objectives, such as minimizing travel time and trip cost, while maximizing the comfortability and reliability, thus **planning a journey** also involves sequential decisions along the trip [18, 19]. Moreover, the design of **wind turbine systems** is also focused on multiple objectives, including more power production and lower fatigue loads, which leads to a trade-off between power production and accumulated damage. In addition to the above instances, recent years have seen multi-objective reinforcement learning applied in wide range of research domains including: recommended systems, transportation management, bidding and pricing and intelligent manufacturing [20, 21, 22, 23]. These various applications are best addressed with multi-objective decision making methods.

5. Discussions and Future Trends

The multi-objective reinforcement learning techniques have been widely adopted in real-world applications. To this end, in this paper, we identify a range of factors that need to be considered when modelling a multi-objective

decision making problem, as well as the traditional and emerging reinforcement learning algorithms to tackle this issue. In addition, we have provided examples on how reinforcement learning can be applied to multi-objective tasks. We hope this article will inspire the future growth of multi-objective reinforcement learning.

In the past few years, reinforcement learning has become more and more powerful and important in dealing with complicated managerial scenarios, and algorithms with good improvements and convergence have been gradually proposed. The Meta-RL [24], Inverse-RL [25] and Transfer-RL [26] have been introduced to simulate multiple agents and fit various environment. In the future, the multi-objective reinforcement learning will continue to improve with this trend, to adapt to more practical application scenarios.

Acknowledgements

The authors would like to thank the anonymous reviewers for valuable suggestions to the improvement of this paper. This work was partially supported by the projects of National Natural Science Foundation of China (No. 71932008, No. 72231010). This work was also supported by China Scholarship Council (No. 202204910376).

References

[1] V Haimes. (1983). MULTIOBJECTIVE DECISION MAKING. NORTH-HOLLAND.

[2] Horn, J. , Nafpliotis, N. , & Goldberg, D. E. . (1994). A Niched Pareto Genetic Algorithm for Multi-Objective Optimization. Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence. Proceedings of the First IEEE Conference on. IEEE.

[3] Deb, K. , Agrawal, S. , Pratap, A. , & Meyarivan, T. . (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: nsga-ii. International Conference on Parallel Problem Solving from Nature.

[4] Barrett, L., & Narayanan, S. (2008). Learning all optimal policies with multiple criteria. In: Proceedings of the 25th International Conference on Machine Learning, pp. 41–47.

[5] Wiering, M. A., & De Jong, E. D. (2007). Computing optimal stationary policies for multi-objective markov decision processes. In: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, pp. 158–165. IEEE.

[6] Moffaert, K. V. , Drugan, M. M. , & A Nowé. (2013). Scalarized multi-objective reinforcement learning: Novel design techniques. 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE.

[7] Nguyen, T. T. , Nguyen, N. D. , Vamplew, P. , Nahavandi, S. , Dazeley, R. , & Lim, C. P. . A multi-objective deep reinforcement learning framework. Engineering Applications of Artificial Intelligence, 96.

[8] Chettibi, S. , & Chikhi, S. . (2012). An adaptive energy-aware routing protocol for MANETs using the SARSA reinforcement learning algorithm. 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems. IEEE.

[9] Sprague, N. , & Ballard, D. H. . (2003). Multiple-Goal Reinforcement Learning with Modular Sarsa(0). Proc Eighteenth International Joint Conference on Artificial Intelligence.

[10] Zeng, F. , Zong, Q. , Sun, Z. , & Dou, L. . (2010). Self-adaptive multi-objective optimization method design based on agent reinforcement learning for elevator group control systems. Intelligent Control & Automation. IEEE.

[11] Peters, J. . (2003). Reinforcement Learning for Humanoid Robots-policy gradients and beyond. 3rd IEEE International Conference on Humanoid Robotics, 2003.

[12] Razavian, A. S. , Azizpour, H. , Sullivan, J. , & Carlsson, S. . (2014). CNN Features off-the-shelf: an Astounding Baseline for Recognition. 2014 IEEE conference on computer vision and pattern recognition workshops. IEEE.

[13] Collobert, Ronan, Weston, & Jason. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008. ACM.

[14] Bae, H. S. , Lee, H. J. , & Lee, S. G. . (2016). Voice recognition based on adaptive MFCC and deep learning. IEEE Conference on Industrial Electronics & Applications. IEEE.

[15] Zh, A. , Kpt, A. , St, A. , Xz, A. , Jie, X. , & Cyb, C. . (2021). Multi-objective optimization of the textile manufacturing process using deep-q-network based multi-agent reinforcement learning. Journal of Manufacturing Systems.

[16] Castelletti, A., Pianosi, F., & Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. Water Resources Research, 49(6), 3476–3486.

[17] Castelletti, A., Pianosi, F., & Soncini-Sessa, R. (2008). Water reservoir control under economic, social and environmental constraints. Automatica, 44(6), 1595–1607.

[18] Ramos, G.de.O., da Silva, B.C., Rădulescu, R., Bazzan, A.L.C., & Nowé, A. (2020). Toll-based reinforcement learning for efficient equilibria in route choice. The Knowledge Engineering Review, 35, e8.

[19] Ramos, G.de.O., Rădulescu, R., Nowé, A., & Tavares, A.R. (2020). Toll-based learning for minimising congestion under heterogeneous preferences. In: B. An, N. Yorke-Smith, A. El Fallah Segh- rouchni, G. Sukthankar (eds.) Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), pp. 1098–1106. IFAAMAS, Auckland, New Zealand.

[20] Jin, J., & Ma, X. (2019). A multi-objective agent-based control approach with application in intel- ligent traffic signal system. IEEE Transactions on Intelligent Transportation Systems, 20(10), 3900–3912.

[21] Krasheninnikova, E., García, J., Maestre, R., & Fernández, F. (2019). Reinforcement learning for pricing strategy optimization in the insurance industry. Engineering Applications of Artificial Intelligence, 80, 8–19.

[22] Lacerda, A. (2017). Multi-objective ranked bandits for recommender systems. Neurocomputing, 246, 12–24.

[23] Lepenioti, K., Pertselakis, M., Bousdekis, A., Louca, A., Lampathaki, F., Apostolou, D., Ment- zas, G., & Anastasiou, S. (2020). Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. In: International Conference on Advanced Information Systems Engineering, pp. 5–16. Springer.

[24] Schweighofer, N. , & Doya, K. . (2003). Meta-learning in reinforcement learning. Neural Networks, 16(1), 5-9.

[25] Baker, C. L. , Saxe, R. , & Tenenbaum, J. B. . (2009). Action understanding as inverse planning. Cognition, 113(3), 329-349.

[26] Ramon, J. , Driessens, K. , & Croonenborghs, T. . (2007). Transfer learning in reinforcement learning problems through partial policy recycling. Springer-Verlag.