Model-Based Multi-objective Reinforcement Learning with Unknown Weights

Tomohiro Yamaguchi[1(✉)], Shota Nagahama[1,2P], Yoshihiro Ichikawa[1], and Keiki Takadama[3]

[1] National Institute of Technology, Nara College, Nara, Japan
{yamaguch, ichikawa}@info.nara-k.ac.jp,
[2] Nara Institute of Science and Technology, Nara, Japan
nagahama.shota.nll@is.naist.jp
[3] The University of Electro-Communications, Tokyo, Japan
keiki@inf.uec.ac.jp

Abstract. This paper describes solving multi-objective reinforcement learning problems where there are multiple conflicting objectives with unknown weights. Reinforcement learning (RL) is a popular algorithm for automatically solving sequential decision problems and most of them are focused on single-objective settings to decide a single solution. In multi-objective reinforcement learning (MORL), the reward function emits a reward vector instead of a scalar reward. A scalarization function with a vector of n weights (weight vector) is a commonly used to decide a single solution. The simple scalarization function is linear scalarization such as weighted sum. The main problem of previous MORL methods is a huge learning cost required to collect all Pareto optimal policies. Hence, it is hard to learn the high dimensional Pareto optimal policies. To solve this, this paper proposes the novel model-based MORL method by reward occurrence probability (ROP) with unknown weights. There are two main features. The first feature is that the average reward of a policy is defined by inner product of the ROP vector and the weight vector. The second feature is that it learns ROP in each policy instead of Q-values. Pareto optimal deterministic policies directly form the vertices of a convex hull in the ROP vector space. Therefore, Pareto optimal policies are calculated independently with weights and just once. The experimental results show that our proposed method collected all optimal policies under four dimensional Pareto optimal policies, and it takes a small computation time though previous MORL methods learn at most two or three dimensions.

Keywords: Multi-objective reinforcement learning · Model-based · Average reward · Reward occurrence probability · Reward vector

1 Introduction

This paper describes solving multi-objective reinforcement learning problems where there are multiple conflicting objectives with unknown weights. Reinforcement learning (RL) is a popular algorithm for automatically solving sequential decision problems which is commonly modeled as Markov decision processes (MDPs). Despite numerous

S. Yamamoto and H. Mori (Eds.): HCII 2019, LNCS 11570, pp. 311–321, 2019.
https://doi.org/10.1007/978-3-030-22649-7_25

reinforcement learning methods, most of them focused on single objective settings where the goal of an agent decides a single solution by the optimality criterion. This reinforcement learning methods are classified according to the learning algorithm and the optimality criterion. The former, there are two kinds of learning algorithms whether directly estimating the MDP model or not, one is the *model-based* approach such as real-time dynamic programming (RTDP) [1] and H-Learning [16] which takes a small time complexity but a large space complexity, and another one is the *model-free* approach [18] such as Q-learning. The *model-based* approach starts with directly estimating the MDP model statistically, then calculates the value of each state as V(s) or the quality of each state action pair: (s, a) (is called a rule) Q(s, a) using the estimated MDP to search the optimal solution that maximizes V(s) of each state. In contrast, the *model-free* approach directly learns V(s) or Q(s, a) without estimating the MDP model.

The latter, there are two kinds of optimality criteria whether using a discount factor or not, one is maximizing the sum of the *discounted rewards*, and another one is maximizing the *average reward* without any discount factor [2, 8, 16, 18]. Most previous RL methods are model-free approach with a discount factor since the model-based approach takes the large space complexity.

A multi-objective MDP (MOMDP) is an MDP in which the reward function emits a reward vector instead of a scalar reward. A scalarization function with a vector of n weights (weight vector) is a commonly used to decide a single solution. The simple scalarization function is linear scalarization such as weighted sum. In this paper, we mainly target the weighted sum function for the scalarization function. However, our method can be applied to other scalarization function such as Tchebycheff norm method.

Multi-objective reinforcement learning (MORL) [6, 11, 13, 14] has several methods which can be divided into two main approaches, the scalar combination and Pareto optimization [3]. In the former case, the scalar combination is to find a single policy that optimizes a combination of the rewards. MOMDP and known weights are input to the learning algorithm, then it output a single solution. In the latter case, Pareto optimization is to find multiple policies that cover the Pareto front, which requires collective search for sampling the Pareto set [10].

MOMDP is input to the learning algorithm, then it output a solution set. Note that there are two ways to select a single solution in the set, one is the scalarization with known weight, another one is a user selection.

Most of the state-of-the-art MORL are model-free value-based reinforcement learning algorithms [4, 7, 9, 13] with a main problem that is MORL previous methods incorporate a huge learning cost to collect all Pareto optimal policies. First, they need a sufficient number of executions for each state-action pair (rule) to collect all Pareto optimal policies since they are model-free methods. Secondly, Pareto optimal set is calculated for each V (state) or Q (state, action) since these methods are value-based. Thirdly, when updating V-values or Q-values, Pareto candidates are added or updated as V/Q-value vector, it must keep a large number of candidates until each Pareto optimal set is converged. Therefore, previous MORL methods take large number of calculations to collect Pareto optimal set for each V/Q-value vector. In contrast, model-based MORL can reduce such a calculation cost [19] than model-free MORLs. However, second and third problems as described above are still remained, and this

method is for only deterministic environments. Thus, it is hard to learn high dimensional Pareto optimal policies by previous methods.

To solve these problems, this paper proposes the novel model-based MORL method by reward occurrence probability with unknown weights. Our approach is based on the average reward model-based reinforcement learning [8] and there are two main features to it. The first feature is that the average reward of a policy is defined by inner product of the ROP vector and the weight vector. The second feature is that it learns ROP in each policy instead of Q-values. Pareto optimal deterministic policies directly form the vertices of a convex hull in the ROP vector space. Therefore, Pareto optimalpolicies are calculated independently with weights and just one time.

The key points of our approach are as follows:

(1) Each objective is defined by a reward rule and its unknown weight.
(2) Multi-objective is defined by the fixed reward rule set and the unknown weight vector.
(3) Each policy is assigned to the reward occurrence probability (ROP) vector where n^{th} value is the occurrence probability of n^{th} reward rule of the policy.
(4) In the ROP vector space (rectangular coordinate system), any stationary policy is mapped to a point where coordinates are indicated by its ROP vector.
(5) Optimal deterministic stationary policies with unknown weights form the vertex of convex hull in the ROP vector space.
(6) Average reward of a policy is defined by the inner product of the ROP vector of the policy and the weight vector.
(7) The range of the weight vector of each optimal deterministic stationary policy can be calculated geometrically.
(8) The stochastic learning environment can be learned by standard MDP model identification method and our proposed search method collects all Pareto optimal policies.

The experimental results show that our proposed method collect all optimal policies under four rewards (four dimensional Pareto optimal policies), and it takes a small computation time though previous MORL methods learn at most two or three dimensions.

2 Model-Based Reinforcement Learning

This chapter describes the framework of model-based RL methods that estimate a model of the environment while interacting with it and search for the best policy of its current estimated model under some optimality criterion. Some comparisons have shown model-based RL are much more effective than model-free methods such as Q-learning [1]. Figure 1 shows an overview of the model-based reinforcement learning system. Note that s is an observed state, a is an executed action, and Rw is an acquired reward. At each time step, the learner is in some state s, and the learner may choose any action that is available in state$s$. At the next time step, the environment responds by randomly moving into a new state s', giving the learner a corresponding reward $\boldsymbol{R}(s, a)$. In Fig. 1 the learning agent consists of three blocks which are model identification

block, optimality of policies block and policy search block. The details of these blocks are described in following section. The novelty of our method lies in policy search block which collects all *reward acquisition policies* on an identified mode according to average reward optimality. The detail of this block is described in Sects. 2.4 and 3.3.

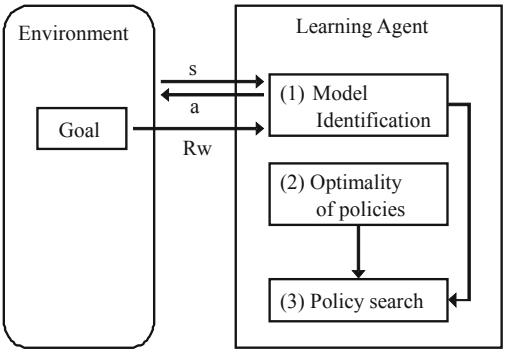

Fig. 1. Framework of model-based reinforcement learning

2.1 MDP Model and Markov Chains

A *Markov decision process* (MDP) [12] in this paper is a discrete time and a discrete state space stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. Note that for a decision maker, actions are the allowing choice, and rewards are giving motivation. A MDP model is defined by of following four elements;

1. Set of states: $\boldsymbol{S} = \{s_0, s_1, s_2, \ldots, s_n\}$
2. Set of actions: $\boldsymbol{A} = \{a_0, a_1, a_2, \ldots, a_n\}$
3. State transition probabilities $\boldsymbol{P}(s'|s, a)$: a probability of occurring state s' when execute action a at state s
4. Reward function $\boldsymbol{R}(s, a)$: an acquired reward when execute action a at state s

State action pair (s, a) is called a *rule*. $\boldsymbol{R}(s, a)$ means that a reward is assigned to a rule (s, a). A *stochastic policy* π is a probability distribution over actions for every possible state. A *deterministic policy* is defined by a function that selects an action for every possible state. This paper mainly deals with *deterministic policies*. $\boldsymbol{P}(s'|s, a)$ means that the probability that the process moves into its new state s' is influenced by the chosen action. Thus, the next state s' depends on the executed rule, that is the current state s and the decision maker's action a, and is independent of all previous executed rules. It is called *Markov property*.

A *Markov chain* is a stochastic model describing a sequence of possible states in which the probability of each state depends only on the previous state. It's an intension of Markov decision processes, the difference is that there is neigher actions nor rewards in a *Markov chain*. This paper focuses on the property that a *policy* of a MDP forms a *Markov chain* of the MDP, it is described later in Sect. 3.4.

We assume MDP model is *ergodic* where the model satisfies these conditions as follows;

1. *irreducible*: All states can be reached from all others.
2. *aperiodic*: Each state is visited without any systematic period.
3. *finite states*: The number of states is finite.

2.2 Model Identification

In model identification block, the state transition probabilities $\boldsymbol{P}(s'|s, a)$ and reward function $\boldsymbol{R}(s, a)$ are estimated incrementally by observing a sequence of (s, a, r). This estimated model is generally assumed by MDP.

Model-based RL methods learn the transition and reward models of the environment by making use of counters that are used in a Maximum-Likelihood Estimation (MLE) to compute approximate transition probabilities and average rewards [19].

Note that the MLE probability is same as the occurrence based probability. Each time the agent selects rule (s, a) and makes a transition to state s', the transition model's counter values $C(s, a)$ and $C(s, a, s')$ are increased by one. In a similar fashion, the obtained reward r is added to the value $R_t(s, a)$ which computes the sum of all rewards obtained by selecting rule (s, a). Finally, the maximum likelihood model of the MDP is computed as Eq. (1).

$$P(s'|s, a) = C(s, a, s')/C(s, a) \text{ and } R(s, a) = R_t(s, a)/C(s, a) \tag{1}$$

2.3 Average Reward Optimality Criterion

Optimality of policies block defines the optimality criterion of the learning policy. In this research, a policy which maximizes average reward is defined as an optimal policy. There are two kinds of optimality criteria on average reward RL, one is gain-optimal which considers acquired rewards only in a stationary cycle, the other is bias-optimal which considers acquired rewards both on a temporally path and the stationary cycle [8]. Equation (2) shows the definition of gain optimal average reward.

$$g^{\pi}(s) \equiv \lim_{N \to \infty} E\left(\frac{1}{N}\sum_{t=0}^{N-1} r_t^{\pi}(s)\right) \tag{2}$$

where N is the number of step, $\mathrm{r}\ {}_{\tau}^{\pi}(s)$ is the expected value of reward that an agent acquired at step t where policy is π and initial state is s and $E()$ denotes the expected value.

2.4 LC-Learning

This section summarizes LC-Learning [5, 15] which is our basic method of policies search. LC-Learning is one of the average reward model-based reinforcement learning

methods. It collects all reward acquisition *deterministic* policies under *unichain* condition. The *unichain* condition requires that every policy in an MDP result in a single *ergodic* class, and guarantees that the optimal average cost is independent of the initial state [17]. The features of LC-Learning are following; (1) Breadth search of an optimal policy started by each reward rule. (2) Calculating average reward using reward acquisition cycle of each policy.

3 Model-Based MORL by Reward Occurrence Probability

3.1 Weighted Reward Vector

To represent multi objective, the reward is divided into d reward types one for each objective, and a weight which represents the importance or preference of that reward is associated with each reward type [4, 10]. In this paper, the reward function is defined by a vector of d rewards (reward vector) $\vec{r} = (r_0, r_1, \ldots, r_{d-1})$ where each r_i represents a position of reward rule and the weight vector $\vec{w} = (w_0, w_1, \ldots, w_{d-1})$ which represents a trade-off among multi objective. A scalarization function with a weight vector is called a *weighted sum of rewards*. Equation (3) shows a weighted sum of rewards defined by inner product of the reward vector $\vec{r} = (r_0, r_1, \ldots, r_{d-1})$ and the weight vector $\vec{w} = (w_0, w_1, \ldots, w_{d-1})$.

$$r(\vec{w}) = \sum_{i=0}^{d-1} w_i r_i = \vec{w} \cdot \vec{r} \tag{3}$$

3.2 Average Reward by Reward Acquisition Probability

Average reward is the expected received rewards per step when an agent performs state transitions routinely according to a policy. A step is a time cost to execute an action. Under an unichain policy π and reward vector $\vec{r} = (r_0, r_1, \ldots, r_{d-1})$, *Reward Acquisition Probability* (ROP) vector $\vec{\boldsymbol{P}}_{\boldsymbol{\pi}}$ is defined as shown in Eq. (4). In that, $\boldsymbol{P_i}$ is the expected occurrence probability per step for reward r_i.

$$\vec{\boldsymbol{P}}_{\boldsymbol{\pi}} \equiv (\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_{\boldsymbol{d}-1}) \tag{4}$$

Average reward ρ_π under a policy π is defined by the inner product of the ROP vector $\vec{\boldsymbol{P}}_{\boldsymbol{\pi}}$ and the weight vector $\vec{w}$ as shown in Eq. (5).

$$\rho_\pi(\vec{w}) = \sum_{i=0}^{d-1} w_i P_i = \vec{w} \cdot 2 \tag{5}$$

3.3 Collecting All Reward Acquisition Policies [15]

Our searching method for reward acquisition policies is based on LC-learning as we described in Sect. 2.4. Deterministic policies which acquire some rewards are searched by converting a MDP into the tree structures where reward acquisition rules are root rule. Figures 2, 3 and 4 shows an illustrated example. The MDP shown in Fig. 2 which consists of four states, six rules and two rewards is converted into the two tree structures shown in Fig. 3. In a tree structure, the path from the root node to the state that is same state to the initial node is the policy. In stochastic environment, some of the rule is deliquesce stochastically. In such case, path from parent node of stochastic rule to the state that is already extracted is part of a policy that contains the stochastic rule. Figure 4 shows all reward acquisition policies in Fig. 2.

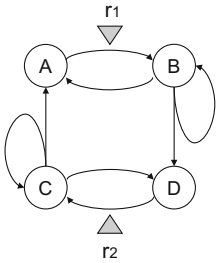

Fig. 2. An example of MDP model

After collecting all reward acquiring policies, for each policy, the state transition probability matrix $\boldsymbol{P}_{\pi}$ of the policy π is prepared as *Markov chain* by estimated **by** MDP model as we described in Sect. 2.2.

3.4 Calculating ROP Vector for Each Reward Acquisition Policy

When the occurrence probability of state i is α_i, it is equivalent to the occurrence probability of the reward rule (i, a) under a deterministic policy π. Calculating method of each ROP vector $\vec{\boldsymbol{P}}_{\pi}$ for each reward acquisition policy π associated with $\boldsymbol{P}_{\pi}$. From the reward acquisition policy set is as follows;

step 1: Set up simultaneous linear equations for each $\boldsymbol{P}_{\pi}$.
Under a deterministic policy π, the occurrence probability vector for all states $\vec{\alpha}_{\pi}$ defined as Eq. (6) is the solution of simultaneous linear equations Eq. (7).

$$\vec{\alpha}_{\pi} \equiv \left(\alpha_0, \alpha_1, \ldots, \alpha_{|S|-1}\right) \tag{6}$$

$$\vec{\alpha}_{\pi}\boldsymbol{P}_{\pi} = \vec{\alpha}_{\pi} \tag{7}$$

step 2: For each $\boldsymbol{P}_{\pi}$, solve Eq. (7) by Gaussian elimination.
step 3: For each $\vec{\alpha}_{\pi}$ derived at step 2, forms ROP vector $\vec{\boldsymbol{P}}_{\pi}$ by picking up the occurrence probability of each reward rule.

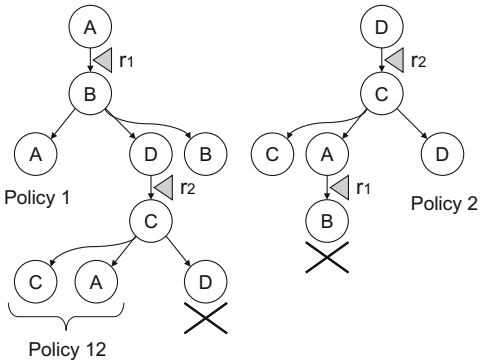

Fig. 3. Searching reward acquiring policies

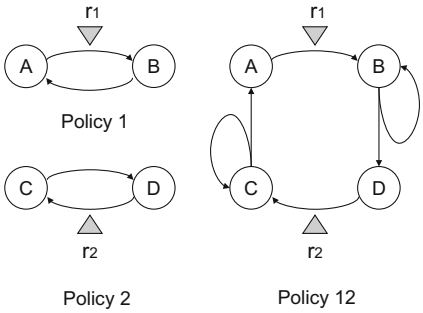

Fig. 4. Three kind of reward acquiring policies

step 4: Make a ROP vector set from the ROP vectors derived at step 3.
step 5: Make a mapping from the reward acquisition policy set to the ROP vector set.

Note that in Eq. (6), α_i is the occurrence probability of state i, the sum of all α_i is 1. In Eq. (7), $\boldsymbol{P}_{\boldsymbol{\pi}}$ is the state transition probability matrix of the policy π since the occurrence probability of each state under a policy forms the *Markov chain* as we described in Sect. 2.1.

3.5 Calculating a Convex Hull from a ROP Vector Set

After collecting all ROP vectors as a set, each ROP vector is located **at** a point in the reward occurrence probability (ROP) vector space. Figure 5 shows an illustrated example of 2 dimension ROP vector space. In Fig. 5, there are two axis, where the horizontal axis is ROP $\boldsymbol{P}_0$, and the vertical axis is ROP $\boldsymbol{P}_1$. For example, the ROP vector of $\vec{\boldsymbol{P}}_{\pi 2}$ is (0.33, 0.33). There are four ROP vectors which are the vertices of the convex hull. We use n dimension convex hull algorithm of free Python library. Note that $\vec{\boldsymbol{P}}_{\pi 1,3}$ is associated with two policies π_1 and π_3.

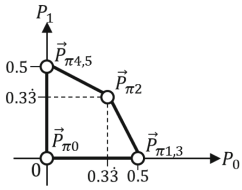

Fig. 5. An illustrated example of 2D ROP vector space with four ROP vectors

The advantage of the proposed method is that Pareto optimal deterministic policies directly form the vertices of a convex hull in the ROP vector space. Therefore, Pareto optimal policies are calculated independently with weights and just one time. Besides, our method can be easily applied **to** other scalarization function such as Tchebycheff norm method in the ROP vector space.

4 Experiments

4.1 Experimental Setup

We conduct the experiment to analyze the bottle neck of our proposed method by evaluating the computation cost of our major processes described in Sect. 3.3 to 3.5.

Experimental conditions on stochastic MDP model as the learning environment are as follows;

(1) The number of states is four cases, 7, 8, 9, and 10 states.
(2) The number of actions is three.
(3) The number of rewards is four.
(4) The transition probability or each rule is setup randomly under the condition that the number of branches of transitions is randomly setup between 1, 2 or 3.

Experimental results are averaged one hundred experiments. Measurement items are Calculation time for each process from Sect. 3.3 to 3.5 as follows;

(1) Section 3.3: Collecting All Reward Acquisition Policies
(2) Section 3.4: Calculating ROP vector for each Reward Acquisition Policy
(3) Section 3.5: Calculating a Convex Hull from a ROP vector set

4.2 Experimental Results

We conducted the learning experiment for one hundred times and each measurement item is an averaged calculation time, its unit is second. In stochastic and ergodic environment, our proposed method successfully collected all reward acquisition deterministic policies including all Pareto optimal policies under a weighted sum scalarization function where weights are unknown. Note that the numbers of stationary cycles of the reward acquisition policies are from 600 (7 states) to 6000 (10 states), the numbers of vertices of the convex hull are from 31 (7 states) to 46 (10 states). Table 1 shows the calculation time for each process of Sects. 3.3 to 3.5 as we described before section.

Table 1 shows that the calculation time of our proposed method is much small under ten states. However, the calculation costs of both (1) Collecting All Reward Acquisition Policies and (2) Calculating ROP vector for each Reward Acquisition Policy seem to be increase exponentially. The increase rate of item (1) is about 10 times and that of item (3) is about 100 times. In contrast, increase rate of the calculation costs of (3) Calculating a Convex Hull from a ROP vector set is about 3 times and is much smaller than (1) and (2).

Table 1. Calculation time for each process of Sects. 3.3 to 3.5

The number of states	(1) Sect. 3.3 [sec]	(2) Sect. 3.4 [sec]	(3) Sect. 3.5 [sec]
7	0.180	0.900	0.00219
8	0.468	4.93	0.00187
9	1.09	32.5	0.00375
10	2.82	207	0.00606

5 Conclusions

This paper proposed the novel model-based MORL method by reward occurrence probability (ROP) with unknown weights and reported our work under the stochastic learning environment with up to ten states, three actions and three or four reward rules. There are two main features. First one is that average reward of a policy is defined by inner product of the ROP vector and the weight vector. Second feature is that it learns ROP in each policy instead of Q-values. Pareto optimal deterministic policies directly form the vertices of a convex hull in the ROP vector space. Therefore, Pareto optimal policies are calculated independently with weights and just once. The experimental results show that our proposed method collected all optimal policies under four dimensional Pareto optimal policies, and it takes a small computation time though previous MORL methods learn at most two or three dimensions.

Future works is to reduce calculation cost for collecting Pareto optimalpolicies. We are planning two ways, one is pruning in the policy search on collecting all reward acquisition policies by estimating the length of the reward acquisition cycle. Another approach is applying parallel computing techniques, multi processing by Multi-core CPU for policy search, and GPGPU for calculating ROP vector.

Achnowledgement. The authors would like to thank Prof. Shimohara and Prof. Habib for offering a good opportunity to present this research. This work was supported by JSPS KAKENHI (Grant-in-Aid for Scientific Research ©) Grant Number 16K00317.

References

1. Barto, A.G., Steven, J., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. Artif. Intell. **72**(1–2), 81–138 (1995)

2. Gao, Y.: Research on Average Reward. Reinforcement Learning. Algorithms, National Laboratory for Novel Software Technology, Nanjing University, 5 November 2006. http://lamda.nju.edu.cn/conf/MLA06/files/Gao.Y.pdf
3. Herrmann, M.: RL 16: Model-based RL and Multi-Objective Reinforcement Learning, University of Edinburgh, School of Informatics (2015). http://www.inf.ed.ac.uk/teaching/courses/rl/slides15/rl16.pdf
4. Hiraoka, K., Yoshida, M., Mishima, T.: Parallel reinforcement learning for weighted multi-criteria model with adaptive margin. Cogn. Neurodyn. **3**, 17–24 (2009)
5. Konda, T., Tensyo, S., Yamaguchi, T.: LC-learning: phased method for average reward reinforcement learning—preliminary results—. In: Ishizuka, M., Sattar, A. (eds.) PRICAI 2002. LNCS (LNAI), vol. 2417, pp. 208–217. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45683-X_24
6. Liu, C., Xu, X., Hu, D.: Multiobjective reinforcement learning: a comprehensive overview. IEEE Trans. Syst. Man Cybern. Syst. **45**(3), 385–398 (2015)
7. Lizotte, D.J., Bowling, M., Murphy, S.A.: Linear fitted-Q iteration with multiple reward functions. J. Mach. Learn. Res. **13**, 3253–3295 (2012)
8. Mahadevan, S.: Average reward reinforcement learning: foundations, algorithms, and empirical results. Mach. Learn. **22**, 159–196 (1996)
9. Van Moffaert, K., Nowe, A.: Multi-objective reinforcement learning using sets of pareto dominating policies. J. Mach. Learn. Res. **15**, 3663–3692 (2014)
10. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: Proceedings of International Conference on Machine Learning (ICML-2005), pp. 601–60 (2005)
11. Pinder, J.M.: Multi-objective reinforcement learning framework for unknown stochastic & uncertain environments. Ph.D. Thesis (2016)
12. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, pp. 385–388. Wiley, New York (1994)
13. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. **48**, 67–113 (2013)
14. Roijers, D.M., Whiteson, S., Vamplew, P., Dazeley, R.: Why multi-objective reinforcement learning? In: European Workshop on Reinforcement Learning, pp. 1–2 (2015)
15. Satoh, K., Yamaguchi, T.: Preparing various policies for interactive reinforcement learning. In: SICE-ICASE International Joint Conference 2006 (2006)
16. Tadepalli, P., Ok, D.: Model-based average reward reinforcement learning. Artif. Intell. **100**, 177–224 (1998)
17. Tsitsiklis, J.N.: NP-hardness of checking the unichain condition in average cost MDPs. Oper. Res. Lett. **35**, 319–323 (2007)
18. Yang, S. Gao, Y., Bo, A., Wang, H., Chen, X.: Efficient average reward reinforcement learning using constant shifting values. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), pp. 2258–2264 (2016)
19. Wiering, M.A., Withagen, M., Drugan, M.M.: Model-based multiobjective reinforcement learning, In: ADPRL 2014: Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, pp. 1–6 (2014)