Multi-Objective Fuzzy Q-Learning to Solve Continuous State-Action Problems

Amirhossein Asgharnia*[1], Howard Schwartz[1], and Mohamed Atia[1]

[1]Department of Systems and Computer Engineering, Carleton university, Ottawa, Canada

Abstract

Many real world problems are multi-objective. Thus, the need for multi-objective learning and optimization algorithms is inevitable. Although the multi-objective optimization algorithms are well-studied, the multi-objective learning algorithms have attracted less attention. In this paper, a fuzzy multi-objective reinforcement learning algorithm is proposed, and we refer to it as the multi-objective fuzzy Q-learning (MOFQL) algorithm. The algorithm is implemented to solve a bi-objective reach-avoid game. The majority of the multi-objective reinforcement algorithms proposed address solving problems in the discrete state-action domain. However, the MOFQL algorithm can also handle problems in a continuous state-action domain. A fuzzy inference system (FIS) is implemented to estimate the value function for the bi-objective problem. We used a temporal difference (TD) approach to update the fuzzy rules. The proposed method is a multi-policy multi-objective algorithm and can find the non-convex regions of the Pareto front.

Keywords: Reinforcement Learning, Differential Games, Q-Learning, Multi-Objective Reinforcement Learning

1 Introduction

Reinforcement learning (RL) is a powerful class of machine learning methods that addresses the learning of sequential actions [1]. RL is different from supervised learning since the learning agent does not have prior knowledge of the proper or desired actions to take. Reinforcement learning tries to maximize a long-term numerical performance index [2]. The numerical performance index is a function of a numerical signal, which is referred to as the *reward signal*. The reward signal rates the learning agent's action given a state. The

*Corresponding author: amirhosseinasgharnia@cmail.carleton.ca

reward signal is calculated via the *reward function* and is given to the agent based on the consequence of taken actions.

There are many studies on choosing and shaping the reward [3–5]. However, all these studies address a scalar reward signal. In other words, the agent will only learn the policy to maximize its long-term performance for only one objective [6]. Whereas, many applications in real-life are of a multi-objective nature [7–10]. For example, increasing the performance of a suspension system while decreasing the control signal [11], a robot that reaches as many targets as possible with the least number of turnings [12], and traffic light controllers to maintain a low fuel consumption, as well as lowering the trip times [13]. In these classes of applications, the performance indicator is not scalar anymore, and it can be defined as a vector [14]. Despite the growing effort in the past decade, there are few studies on problems where the reward signal is a vector, especially if the problem is in the continuous-action and the continuous-state domain.

Multi-objective reinforcement learning can be regarded as a combination of reinforcement learning techniques and multi-objective optimization (MOO) [15]. One of the methods to optimize the MOO problem is to convert the objective functions into one fitness function. It is common to use scalarization methods such as the weighted sum method, the constraint method, the sequential method, and the min-max method [15]. The other standard method in MOO is finding all optimal answers in a single implementation. In the latter method, all solutions, which are non-dominated, are depicted via Pareto fronts. Similar to MOO, the multi-objective reinforcement algorithms are categorized into a single-policy or a multi-policy algorithm. Single-policy algorithms can only find one policy in a single implementation. Thus, the user may know the objective preferences or use methods to change the preferences as a function of time [16]. On the other hand, single-policy algorithms are time efficient and use less computational effort. The second class, multi-policy algorithms, give all optimal policies in the form of the Pareto front in a single implementation [17]. The Pareto front is a suitable tool to show the compromises between different objectives. Although multi-policy algorithms are more complicated and computationally inefficient, they can better show the relationship between the objectives.

There are many algorithms introduced to train agents with a multi-objective reward function. However, most of them work on the discrete domain. Thus, the usage of the MORL algroithms for solving differential games is relying on discretization. The contributions of this paper are as follows:

- We propose a new multi-objective reinforcement learning algorithm, which can operate in the continuous state and the continuous action domain.
- The performance of the proposed methods is depicted in a proposed multi-objective differential game.

This paper proposes a new multi-objective reinforcement algorithm for solving the problems in the continuous-action continuous-state domain. To our best knowledge, it is the first

MORL algorithm in the literature which uses a temporal difference scheme. The application of such a method is for learning control strategies, solving differential games, or any application in which the states and actions are represented in the continuous domain. There are many multi-objective reinforcement algorithms in the literature, which we introduced in section 2. However, they are whether in the discrete action-state domain or based on quantization of the states. Our proposed algorithm is based on the extension of the fuzzy Q-learning (FQL) algorithm [18] to the multi-objective domain. The proposed algorithm is a multi-policy MORL algorithm and approximates the global optimal Pareto front in a single implementation. It is shown that the algorithm can also achieve the solutions that are in the non-convex region of the Pareto front.

This paper is organized in six sections. Section 2 presents a brief literature review. In section 3 we address the problem as well as the classical fuzzy Q-learning algorithm. Section 4 proposes the multi-objective FQL algorithm. In section 5, we present simulations and comparisons. Finally, section 6 concludes the paper and presents the future studies.

2 Background

The earlier studies of multi-objective reinforcement learning go back to multiple objective decision making. In [7], a multi-criterion decision-making process for finding the shortest path is proposed. In addition, there are a few studies on multi-objective dynamic programming that were proposed in early 80s [19].

A multi-criteria reinforcement learning (MCRL) algorithm is proposed in [14] in 2002. In [14], the problem is modelled in a multi-agent structure: a player is playing against the non-stationary moves of the environment, modelled as an adversarial player. The proposed method is studied in more depth in [20].

Scalarization of the reward function is a straightforward task if the designer knows the preferences of the reward elements. For instance, in [21], the reward function has two components. The authors could find a set of weights to scalarize the reward function with the weighted sum method. However, the reward weights were stationary during the learning process. A time-varying reward weight for scalarization is proposed in [16].

The scalarization methods in reinforcement learning are criticized in [9] for their deficiency in obtaining the non-convex parts of the Pareto front. The researchers address the limitation of converting a MOO problem to an SOO problem. Although algorithms such as the one proposed in [16] decrease the solution complexity, the non-convex regions of the Pareto front cannot be discovered. Authors in [9] provide three standard benchmarks for researchers to test their proposed MORL algorithms.

A MORL algorithm is proposed to find all optimal policies in [17], which is called the convex hull value iteration algorithm. The algorithm still suffers from not being able to find the non-convex regions of the Pareto front. However, the article moved the multi-objective reinforcment learning to another level. The convex hull value iteration algorithm is able to find all possible policies for any combination of reward weights in a single run by extending the definition of the Bellman equation to a multi-objective domain.

Empirical evaluation metrics for the performance and limitations of the MORL algorithms are addressed in [22]. Performance metrics for both multi-policy and single-policy MORL algorithms are proposed. The paper gives three benchmarks as well as their known Pareto fronts for researchers to compare their proposed algorithms. It is noted that for multi-policy algorithms, the metrics must show how well the algorithm approximates the Pareto front and how fast the algorithm finds the polices.

The exploration-exploitation dilemma is a challenging problem in the reinforcement learning algorithms. In single-objective reinforcement learning, exploration-exploitation strategies such as ε-greedy and softmax are well-studied. However, ε-greedy and softmax cannot be implemented in a multi-objective algorithm without modification. The hypervolume indicator is proposed as a metric to assess the quality of the action selection strategy in [23]. The researchers combined the idea of reinforcement learning and multi-objective optimization to develop a new algorithm called hypervolume-based MORL (HB-MORL).

Until 2014, most of the studies on MORL were on single-policy algorithms. However, the number of studies on multi-policy MORL has rapidly increased. Authors in [15] give a comprehensive overview of the research path from the beginning of MORL to 2014.

A Q-learning based multi-objective reinforcement algorithm is proposed in [24]. The algorithm can discover the Pareto front for finite episodic games. In addition, it was able to find the non-convex regions of the Pareto front. The authors compared three exploration mechanisms, discovering that the hypervolume ε-greedy method can outperform other suggested methods.

Learning a continuous approximation for the Pareto front via the gradient method is studied in [25]. The researchers proposed a method called the Policy Manifold Gradient Algorithm (PMGA), which returns a parametric function of a manifold that is an approximation for the local Pareto front. The function maps the objective space to the expected return. Unlike studies [14, 20], the proposed method does not need to use n optimizers to generate n trajectories to the Pareto front. Indeed, the learning is conducted in a single gradient ascent run. A deeper study on the proposed algorithm is presented in [26].

In [27], the researchers combined a deep Q-learning network (DQN) with a multi-objective problem. The first study used the DQN to solve the MOMDP with a high-dimensional input set. They implement an idea similar to [16] in scalarization. Thus, the proposed algorithm can handle high-dimensional inputs without knowing the objective preferences. On the other hand, the algorithm is unable to find the non-convex parts of the Pareto front.

A softmax exploration strategy is proposed for MORL algorithms in [28]. They implement the modified exploration mechanisms common in single-objective problems and study the method's effectiveness in multi-objective cases.

A temporal difference approach in multi-objective Q-learning, called MPQ-learning, is studied in [29]. The study addresses the algorithm in a discrete state domain. They illustrate that the algorithm is effective as a scalarization-based method. In addition, MPQ-learning can find the convex regions of the Pareto front.

A modification to the algorithm that was proposed in [24] is presented in [30]. The proposed algorithm uses a multi-agent preference to find the Pareto front. The paper tries

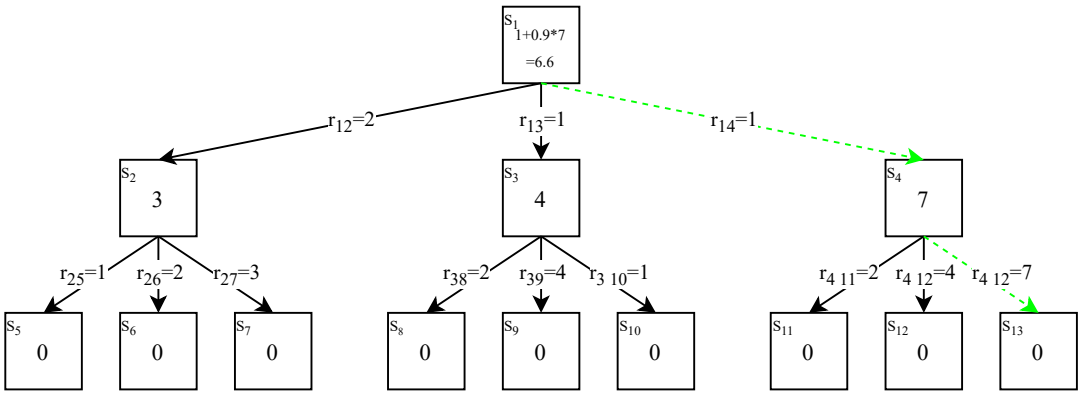

Figure 1: Example 1. An imaginary example for an action-state scenario where rewards are scalars

to address the deficiency in the earlier algorithm in finding the global policy by using several searching agents.

2.1 Difference Between the SORL algorithms and the MORL algorithms

The primary difference between an MDP and a MOMDP is the reward. In a SORL algorithm, which is a solution method for the MDP, the reward signal is a scalar ($[r_t]_{1\times 1}$), whereas, in a MORL, which is a solution method for MOMDP, the reward signal is a vector $\vec{r}_t = [r_t^1, r_t^2, \cdots, r_t^n]^T$, where n is the number of objectives. As a result, in MORL algorithms, the Q-values associated with each action, given each state, is a set of non-dominated vectors. In a SORL, the reward function in a scalar, so the associated Q-value for an action given a state is a scalar. We provide two examples to show why the Q-values become a set of vectors. Example 1 shows the state-value calculation for a discrete game, where reward signals are scalars. Example 2 illustrates the state-value calculation for a discrete game, where reward signals are vectors.

Example 1: This example shows an episodic game with three steps as in Figure 1. In each step, the agent can take three different actions. The game terminates after taking three actions. Thus, in the states after the second action, the value-functions are zero ($V(S_5) = V(S_6) = \cdots = V(S_{13}) = 0$). Each action, given each state, returns a reward r_{ab}, which means the reward of going from state a to state b. The Bellman equation is used to find the optimal value-functions, which is as follows for $\gamma = 0.9$,

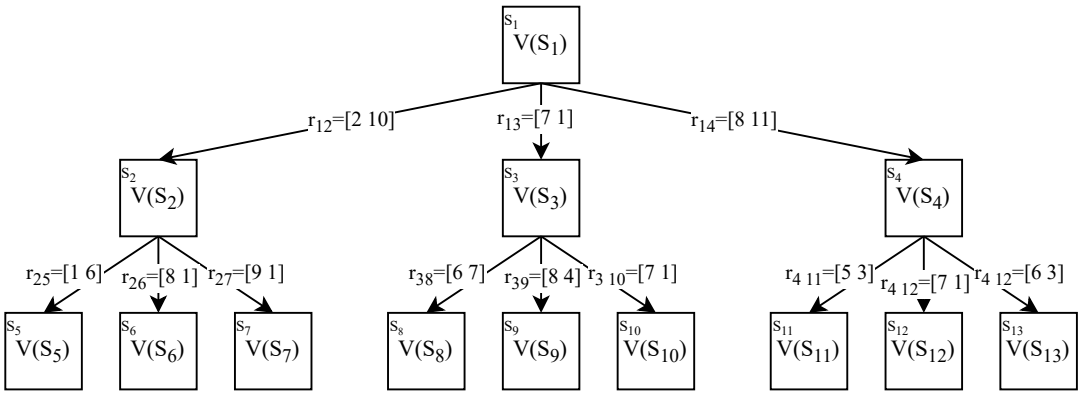

Figure 2: Example 2. An imaginary example for an action-state scenario where rewards are vectors

$$
\begin{aligned}
&V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S,\\
&V^*(s) = \max(r_{t+1} + \gamma V^*(s_{t+1})),\\
&V^*(s_2) = \max(r_{25} + \gamma V^*(s_5), r_{26} + \gamma V^*(s_6), r_{27} + \gamma V^*(s_7)) = \max(1, 2, 3) = 3,\\
&V^*(s_3) = \max(r_{38} + \gamma V^*(s_8), r_{39} + \gamma V^*(s_9), r_{310} + \gamma V^*(s_{10})) = \max(2, 4, 1) = 4,\\
&V^*(s_4) = \max(r_{411} + \gamma V^*(s_{11}), r_{412} + \gamma V^*(s_{12}), r_{413} + \gamma V^*(s_{13})) = \max(2, 4, 7) = 7,\\
&V^*(s_1) = \max(r_{12} + \gamma V^*(s_2), r_{13} + \gamma V^*(s_3), r_{14} + \gamma V^*(s_4)) = \max(4.7, 4.6, 7.3) = 7.3.
\end{aligned}
$$

Thus, the greedy policy is shown in Figure 1 with the dash line.

Example 2: In this example, the rewards are 1×2 vectors. Let us calculate $V^*(S_2)$, $V^*(S_3)$, and $V^*(S_4)$. In this scenario, a 1×2 reward vector cannot be added to a 1×1 value-function. Thus, the value-functions of states 5 to 13 are assumed as [0, 0] instead of being 0 in Example 1. In other words,

$$V^*(s_5) = V^*(s_6) = \cdots = V^*(s_{13}) = [0,\ 0].$$

If we use the same method as in Example 1, the value-functions will be as follows,

$$
\begin{aligned}
&V^*(s_2) = \max(r_{25} + \gamma V^*(s_5), r_{26} + \gamma V^*(s_6), r_{27} + \gamma V^*(s_7)) = \max([1,\ 6], [8,\ 1], [9,\ 1]),\\
&V^*(s_3) = \max(r_{38} + \gamma V^*(s_8), r_{39} + \gamma V^*(s_9), r_{310} + \gamma V^*(s_{10})) = \max([6,\ 7], [8,\ 4], [7,\ 1]),\\
&V^*(s_4) = \max(r_{411} + \gamma V^*(s_{11}), r_{412} + \gamma V^*(s_{12}), r_{413} + \gamma V^*(s_{13})) = \max([5,\ 3], [7,\ 1], [6,\ 3]),
\end{aligned}
$$

where, the max() operator cannot be used in this situation. Vectors like [1, 6] and [9, 1] are incomparable and $\max([1,\ 6], [9,\ 1])$ is meaningless. Instead of looking for a maximum, we will find the non dominated solutions. In order to calculate the value-functions, we modify the Bellman equation into a multi-objective domain. Our approach is similar to the Bellman

equation that was used in [24] and was mentioned in (14). The Bellman equation is defined as follows,

$$V^*(s) = \text{ND}(\bigcup_{s' \in S}(\vec{r} + \gamma V^*(s'))), \tag{1}$$

where ND() is the non-dominated sorting operator. The operator keeps the non-dominated vectors and deletes the dominated. With (1), the value-functions will be as follows,

$$\begin{aligned}
V^*(s_2) &= \text{ND}\bigcup(r_{25} + \gamma V^*(s_5), r_{26} + \gamma V^*(s_6), r_{27} + \gamma V^*(s_7)) \\
&= \text{ND}\bigcup([1,\ 6], [8,\ 1], [9,\ 1]) = [[1,\ 6], [9,\ 1]]^T, \\
V^*(s_3) &= \text{ND}\bigcup(r_{38} + \gamma V^*(s_8), r_{39} + \gamma V^*(s_9), r_{310} + \gamma V^*(s_{10})) \\
&= \text{ND}\bigcup([6,\ 7], [8,\ 4], [7,\ 1]) = [[6,\ 7], [8,\ 4]]^T, \\
V^*(s_4) &= \text{ND}\bigcup(r_{411} + \gamma V^*(s_{11}), r_{412} + \gamma V^*(s_{12}), r_{413} + \gamma V^*(s_{13})), \\
&= \text{ND}\bigcup([5,\ 3], [7,\ 1], [6,\ 3]) = [[7,\ 1], [6,\ 3]]^T.
\end{aligned}$$

Thus, we can calculate $V^*(s_1)$ as follows,

$$\begin{aligned}
V^*(s_1) &= \text{ND}\bigcup(r_{12} + \gamma V^*(s_2), r_{13} + \gamma V^*(s_3), r_{14} + \gamma V^*(s_4)), \\
&= \text{ND}\bigcup([2,\ 10] + 0.9[[1,\ 6], [9,\ 1]]^T, [7,\ 1] + 0.9[[6,\ 7], [8,\ 4]]^T, [8,\ 11] + 0.9[[7,\ 1], [6,\ 3]]^T), \\
&= \text{ND}\bigcup([[2.9,\ 15.4], [10.1,\ 10.9]]^T, [[12.4,\ 7.3], [14.2,\ 4.6]]^T, [[14.3,\ 11.9], [13.4,\ 11.9]]^T), \\
&= [[2.9,\ 15.4], [10.1,\ 10.9], [12.4,\ 7.3], [14.3,\ 13.7]]^T,
\end{aligned}$$

which show that the value-functions can be a set of vectors. The same results can happen for the FQL algorithm in section 3.1. In section 3.1, the Q-value corresponding to each action for each rule was a scalar. However, in the multi-objective case, the Q-value corresponding to each action for each rule will be a set of non-dominated vectors.

3 Fuzzy Q-Learning and Multi-Objective Games

We review the classical fuzzy Q-learning (FQL) algorithm. In addition, we introduce a new game to study the effectiveness of multi-objective reinforcement learning algorithms in continuous state space.

3.1 Fuzzy Q-Learning

The fuzzy Q-learning (FQL) algorithm was proposed in [18] as an extension to the ordinary Q-learning. The Q-values in the FQL algorithm are stored in a Takagi-Sugeno fuzzy inference system (FIS) instead of Q-tables. Thus, the algorithm can handle games with continuous state space like differential games. The FQL algorithm generates continuous actions. However, the output parameters of the fuzzy logic controller (FLC) are selected from a discrete action set $A = \{a_1, a_2, \cdots, a_n\}$. Since a Takagi-Sugeno FLC is used, the output parameter of the fuzzy inference system is the linear combination of action set members. In a differential game, the FLC generates the control signal. The control signal (U) is computed as follows,

$$U(\tilde{x}) = \sum_{l=1}^{M} \Phi^l a^l, \tag{2}$$

where $\tilde{x} = (x_1, x_2, \cdots, x_n)$ is the input set, M is the total number of fuzzy rules, a^l is the action for given rule l and can be viewed as the output parameter. The term Φ^l is the firing strength for rule l, which is defined as follows,

$$\Phi^l = \frac{\prod_{l=1}^{n} \mu^{F_j^l(x_i)}}{\sum_{l=1}^{M}(\prod_{l=1}^{n} \mu^{F_j^l(x_i)})}, \tag{3}$$

where $\mu^{F_j^l(\tilde{x})}$ is the membership degree of the ith input in the jth membership function for the lth rule. To find the best a^ls for (2), the FQL algorithm is implemented. In the FQL algorithm, a Q-value is assigned to each action for each rule. The greedy action selection in the FQL is selecting the action with the highest Q-value for each rule in the fuzzy inference system.

The training process starts with the implementation of a suitable mechanism for the exploration-exploitation dilemma. To this end, to simulate an episode in the learning process, a fraction of a^ls in (2) are selected randomly. We can use the *Softmax* function for selection. Thus, the action with a higher Q-value has a higher chance of being chosen. For each rule, the probability of choosing action a_i for rule l is as follows,

$$\Pr(a_i^l) = \frac{exp(\tau q(l, a_i))}{\sum_{j=1}^{|A|} exp(\tau q(l, a_j))}, \tag{4}$$

where $\Pr(a_i^l)$ is the probability of choosing action a_i for rule l, $q(l, a_i)$ is the Q-value of action a_i given rule l, $|A|$ is the cardinality of the action set $A = \{a_1, a_2, \cdots, a_n\}$, and τ is the softmax temperature. Action selection is conducted once for a time step.

An alternative to softmax function is the ε-greedy method. In the ε-greedy strategy, a random action is selected with the probability of ε given each rule, and the action, corresponding to the largest Q-value is selected with the praobability of $1-\varepsilon$. Eq. (5) shows the method.

$$a^l = \begin{cases} \text{random action from } A \text{ with probability } \epsilon \\ \arg\max_{a \in A}(q(l,a)) \text{ with probability } 1-\varepsilon \end{cases} . \tag{5}$$

After an output parameter is selected for each rule, the control signal is calculated for each time step as follows,

$$U(\tilde{x}_t) = \sum_{l=1}^{M} \Phi_t^l a_t^l, \tag{6}$$

where $U(\tilde{x}_t)$ is the control signal at time step t, and Φ_t^l is the firing strength of rule l at time step t. The term a_t^l is the selected action for rule l at time step t. The agent takes the action, moves to the new state, and receives the reward. The global Q-function is calculated as follows,

$$Q_t(\tilde{x}_t) = \sum_{l=1}^{M} \Phi_t^l q_t(l, a_t^l), \tag{7}$$

where $Q_t(\tilde{x}_t)$ is the global Q-function for input $\tilde{x}$ at time step t. The term $q_t(l, a_t^l)$ is the Q-value of action a_t^l, given rule l at time step t. The global Q-function with the maximum Q-values is calculated as follows,

$$Q_t^*(\tilde{x}_t) = \sum_{l=1}^{M} \Phi_t^l \max_{a \in A} q_t(l,a), \tag{8}$$

where $\max_{a \in A} q_t(l,a)$ is the maximum Q-value among all actions given rule l. By having (7) and (8), the temporal difference is calculated as follows,

$$\tilde{\varepsilon}_{t+1} = r_{t+1} + \gamma Q_t^*(\tilde{x}_{t+1}) - Q_t(\tilde{x}_t), \tag{9}$$

where γ is the discount factor, r_{t+1} is the received reward at time step $t+1$. The Q-values are updated in each time step as follows,

$$q_{t+1}(l, a_t^l) = q_t(l, a_t^l) + \alpha \tilde{\varepsilon}_{t+1} \Phi_t^l, \tag{10}$$

where α is the learning rate.

3.2 Reach-Avoid Game

A single agent game with two conflicting reward functions is proposed in this paper. A robot wants to reach a target (T) in the proposed game while avoiding a pit (P). The game is a differential game since the robot motion equations are described with differential equations as follows,

$$\begin{cases} \dot{x} = v \cdot cos(\theta) \\ \dot{y} = v \cdot sin(\theta) \\ \dot{\theta} = \frac{U \cdot v}{L} \end{cases} . \tag{11}$$

where (x,y) is the agent's location in the field, θ is the angle between the heading and the x-axis, U is the steering angle of the robot, as well as the output of the controller, v is the robot's speed, and L is the axle length.

The game states are defined with three parameters, as follows,

$$\text{Inputs} = [\alpha_{AP} \quad \alpha_{AT} \quad d_{AT}]. \tag{12}$$

In (12), α_{AP} is the angle between the agent's heading and a straight line between the agent and the pit, α_{AT} is the angle between the agent's heading and a straight line between the agent and the target. The term d_{AT} is the distance between the agent and the target. The three inputs in (12) are sufficient for defining the state of the system if the target and pit locations remain unchanged during the learning process. Figure 3 depicts the game, as well as the inputs.

The proposed game has two conflicting objectives. The agent has to decrease its distance to the target. The agent has to increase its distance from the pit. The objectives are defined by the rewards as follows,

$$\begin{cases} R_1 = d_{AT}(t+1) - d_{AT}(t) \\ R_2 = d_{AP}(t) - d_{AP}(t+1) \end{cases} , \tag{13}$$

where R_1 and R_2 are the reward signals, $d_{AT}(t)$ is the distance between the agent and target in time step t, and $d_{AP}(t)$ is the distance between the agent and the pit in time step t.

4 Multi-Objective Fuzzy Q-Learning (MOFQL)

This section presents our multi-objective reinforcement algorithm, called the multi-objective fuzzy Q-learning (MOFQL) algorithm. The proposed algorithm is an extension to the classical FQL algorithm, which was described in section 3.1. The proposed algorithm is multi-policy; therefore, it can find multiple optimal non-dominated policies simultaneously. On the other hand, the proposed algorithm can find the non-convex regions of the Pareto front.

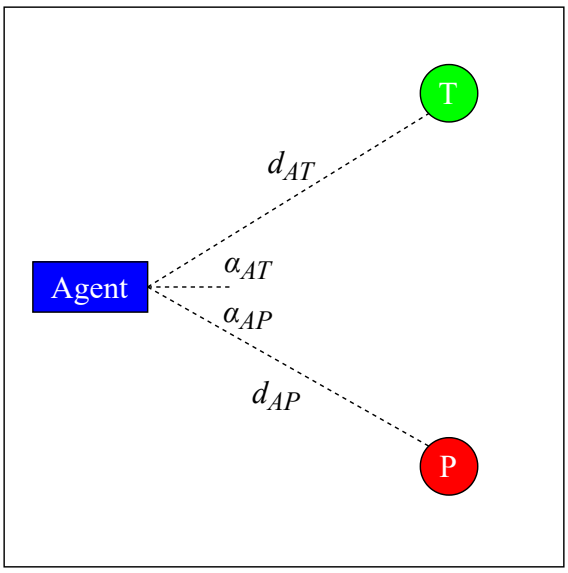

Figure 3: Reach-Avoid game with a target and a pit and one agent

4.1 Preliminaries

We describe some of the preliminaries and definitions, which are used in our algorithm.

4.1.1 Bellman Equation

In the proposed MOFQL algorithm, we use the extended Bellman equation definition proposed in [24] as a premise for our definition. The Bellman equation used in [24] is as follows,

$$\hat{Q}_{set}(s,a) = \boldsymbol{R}(s,a) \oplus \gamma \mathrm{ND}_t(s,a), \tag{14}$$

where, $\hat{Q}_{set}(s,a)$ is the Q-value of state s and action a, $\boldsymbol{R}(s,a)$ is the vector reward of action a in state s. The operator $\oplus$ adds a vector v to a set of vectors V as follows,

$$v \oplus V = \bigcup_{v' \in V} (v + v'). \tag{15}$$

Unlike [24], we propose a temporal difference (TD) scheme to find the optimal policies.

4.1.2 Non-dominated Q-values

In the MOFQL algorithm, the Q-value given each rule and action is a set of non-dominated vectors as demonstrated in (16).

$$\boldsymbol{q}(l,a) = \begin{bmatrix} q_1^1 & \cdots & q_1^n \\ \vdots & \ddots & \vdots \\ q_m^1 & \cdots & q_m^n \end{bmatrix}_{m\times n}, \tag{16}$$

where, $\boldsymbol{q}(l,a) : \Re^2 \leftarrow \Re^{m\times n}$ and n is the number of the objectives, and m is the number of non-dominated Q-value assigned to action a, given rule l. This paper addresses a bi-objective algorithm; thus $n = 2$. The Q-value of rule l and action a for an imaginary policy may look as follows,

$$\boldsymbol{q}(l,a) = \begin{bmatrix} q_1^1 & q_1^2 \\ q_2^1 & q_2^2 \\ q_3^1 & q_3^2 \end{bmatrix}_{3\times 2}, \tag{17}$$

where $\boldsymbol{q}(l,a)$ is a bi-objective Q-value with three non-dominated members. In (17), the matrix elements are scalars. We define the Bellman equation as in (1).

4.1.3 Global Q-function

In a single-objective FQL, the global Q-function maps the states into a scalar to show the quality of the state. As shown in (7), by choosing the maximum Q-value of each action, given each rule. In a multi-objective the definition of maximum Q-value changes to non-dominated Q-values. Since usually there are more than one non-dominated Q-value given each action, in a multi-objective FQL, there would be more than one global Q-function. Thus, the Bellman equation must be used multiple times to take all global Q-functions into account.

4.2 Exploration-Exploitation

Action selection strategies in a multi-objective reinforcement learning algorithm are addressed in [24] and [28]. The ε-greedy and softmax function methods cannot be implemented in MORL without augmentation. Thus, in [24], three alternative approaches are presented and their performances are compared. The first exploration-exploitation strategy is the use of the hypervolume indicator. The hypervolume maps the quality of a Pareto front into a scalar [31]. To use the hypervolume method, the hypervolume of the non-dominated members of an action given a state are computed [23]. The action with the highest hypervolume is expressed as the greedy action. The other method in [24] is called cardinality evaluation. In cardinality evaluation, the action with the highest number of members in the Pareto front is considered as the greedy action. The third exploration-exploitation approach in [24] is the

Pareto set evaluation. In the third method, the actions that have members in the Pareto front have a higher chance of being selected by the agent. It is shown that the hypervolume method outperforms other approaches by having a better convergence rate, as well as collecting the highest reward in the end. In addition to the mentioned approaches, thresholded lexicographic ordering (TLO) is implemented in [28]. TLO evaluates the Q-values of each action, given each state. In this paper, we used the hypervolume method for the exploration-exploitation dilemma. The action selection process is done at each time step. At each time step, the softmax function calculates the probability of selecting each action.

The hypervolume is the volume (area in a bi-objective problem) surrounded by the points in the Pareto front and a reference point chosen by an external source. Figure 4 (a) illustrates the hypervolume for a bi-objective problem. In Figure 4 (a) the reference point is (0,0). In the hypervolume method, the action with a higher hypervolume is more likely to be selected. In [24], the ε-greedy method was used, where the action with the highest hypervolume is chosen with probability of $1-\varepsilon$ and a random action is chosen with the probability of ε. In this paper, we use the *softmax* function to find the likelihood of each action being selected. The probability of action j, being selected as the output parameter of rule l is shown as follows,

$$\Pr(a_l^j) = \frac{\exp(\tau \cdot c_l^j)}{\sum_{i=1}^{|A|} \exp(\tau \cdot c_l^i)}, \tag{18}$$

where, A is the action set, $|A|$ means the cardinality of A. The term τ is the softmax temperature. The term c_l^i is the normalized hypervolume of the action i, given rule l and is calculated as follows,

$$c_l^i = \frac{HV_l^i + 2|\min(HV_l^i)| + 0.01}{\sum_{i=1}^{|A|}(HV_l^i + 2|\min(HV_l^i)| + 0.01)}. \tag{19}$$

In (19), HV_l^i is the hypervolume of the action i, given rule l.

Figure 4 (b) illustrates the hypervolume of the Q-values for four actions, given a single rule, $\boldsymbol{q}(l, a)$. The hypervolume for each action is shown in a distinct colour for a common reference point. The reference point is at (0,0). Figure 4 (b) shows the actions whose Q-values are in the middle (actions b and c) have a higher hypervolume. In contrast, the actions in the boundaries have less hypervolume (actions a and d). Thus, the probability of actions b and c being selected is more than actions a and d. Table 1 shows the hypervolume of the actions in Figure 4, their normalized value, and the output of the probability of each action being selected for $\tau = 1.0$ in (18).

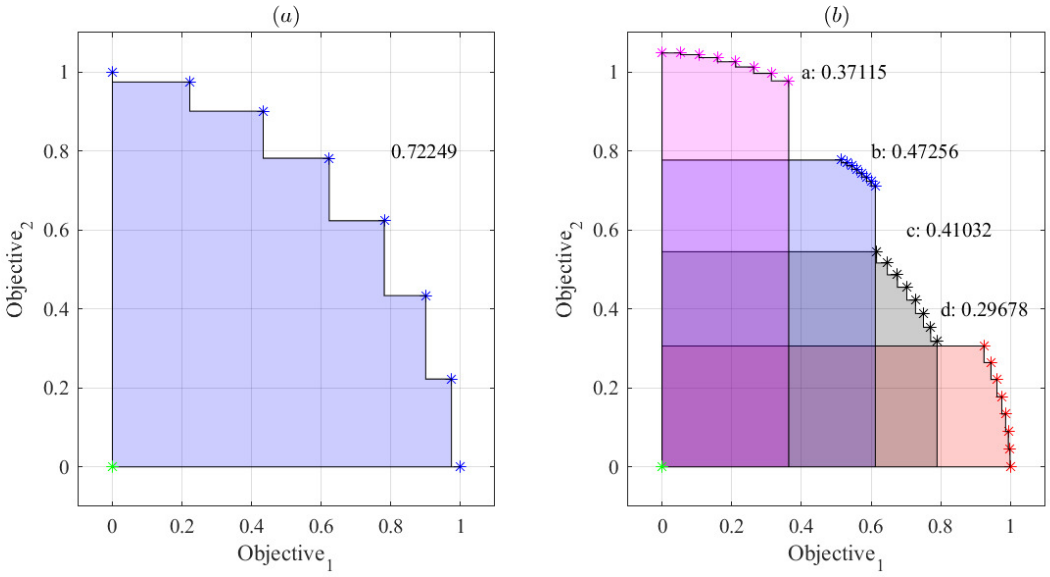

Figure 4: The hypervolume. (a) The hypervolume of a set of non-dominated Q-values in 2-Dimensional domain. (b) The hypervolume of four actions.

Table 1: The hypervolume of the actions in Figure 4, their normalized value, and the output of the probability of each action being selected for $\tau = 1.0$.

Action	Hypervolume	Normalized hypervolume	Probability based on hypervolume
a	0.37115	0.24581	0.24779
b	0.47256	0.27138	0.26079
c	0.41032	0.25569	0.25173
d	0.29698	0.22712	0.23869

4.3 Calculating The Global Q-Function

The output parameters of the FLC are selected by using the hypervolume method in section 4.2. Since a softmax function is implemented as in (19), the output parameters are a combination of exploitation and exploration. The FLC returns an action by computing the agent's state as in (2). The agent takes an action, moves to a new state, and receives a vector reward. As in the single-objective FQL, shown in section 3.1, a global Q-function must be calculated. In the single-objective FQL, the global Q-function was calculated by substituting the maximum Q-value of each action given a rule as the output parameters as in (8). In a multi-objective FQL, we have a set of non-dominated Q-values for each action, given each rule. Thus, any arbitrary Q-values on the Pareto front can be used to calculate the global Q-function. Ideally, all members of the Pareto front must be used to calculate multiple global Q-functions. However, such a process has a huge computational burden. We propose to extract several but a limited number of global Q-functions. We note the number of selected global Q-function as the parameters H. The difference between the calculated global Q-function is the influence of each objective. For instance, one of the global Q-functions only considers the influence of the first objective, one of them considers the influence of the second objective, and the rest are divided among them. First, we define G_l^* as the non-dominated union of all $\boldsymbol{q}(l,a)$s for all $a \in A$, given rule l as follows,

$$G_l^* = \mathrm{ND}(\bigcup_{a \in A} \boldsymbol{q}(l,a)), \quad l = 1, 2, \cdots, M. \tag{20}$$

In (20) G_l^* is a $m \times n$ matrix, where n is the number of objectives, which is two in this study. The term m is the number of non-dominated Q-values.

Example 3: Let us assume that we have Q-values of three actions, given one rule as follows,

$$\boldsymbol{q}(l,a_1) = \begin{bmatrix} 10 & 1 \\ 9 & 3 \\ 5 & 4 \end{bmatrix}$$
$$\boldsymbol{q}(l,a_2) = \begin{bmatrix} 6 & 1 \\ 8 & 3 \\ 2 & 4 \end{bmatrix}$$
$$\boldsymbol{q}(l,a_3) = \begin{bmatrix} 9 & 7 \\ 8 & 8 \end{bmatrix},$$

where, the term G_l^* will be as follows,

$$G_l^* = \text{ND}(\bigcup_{i=1}^{3} \boldsymbol{q}(l, a_i)) = \text{ND}(\begin{bmatrix} 10 & 1 \\ 9 & 3 \\ 5 & 4 \\ 6 & 1 \\ 8 & 3 \\ 2 & 4 \\ 9 & 7 \\ 8 & 8 \end{bmatrix}) = \begin{bmatrix} 10 & 1 \\ 9 & 7 \\ 8 & 8 \end{bmatrix}.$$

The union of Q-values of all the actions given rule l is illustrated in Figure 5 (a). Figure 5 (b) shows the **non-dominated** union of Q-values of all the actions given rule l. The non-dominated Q-values on Figure 5 (b) are the elements of G_l^*. Now, the goal is to select some of G_l^* members uniformly to calculate the global Q-functions. To do so, we calculate the angles of each Q-value with respect to the x-axis and a reference point. The process is shown in Figure 5 (b), where the colour points are the points defined by the rows of G_l^*'s matrix. Each colour represents an action in the action set $A = \{a_1, a_2, \cdots, a_m\}$, where m is the number of actions. There are H dash lines and their angles with x-axis are uniformly distributed in the interval of $[0, \frac{\pi}{2}]$. We refer to each dashed line by its angle as θ_j $(j = 1, \cdots, H)$. The angle of each dashed line with the x-axis indicates the influence of each objective. For instance, the dashed line with $\theta = 0$, only considers the influence of the first objective. The dashed line with $\theta = \frac{\pi}{2}$ only considers the influence of the second objective. The dashed line with the slope of θ will be as follows,

$$\frac{\text{Objective 1}}{\text{Objective 2}} = \tan(\theta). \tag{21}$$

The number of dash lines is set at the beginning of the algorithm. For each dashed line, given rule l, the closest Q-value on the Pareto front to the dash lines is selected. In Figure 5 (b), selected Q-values are shown with green circle. We refer to the selected Q-values as $G_{l\theta_j}^*$. The process is repeated for each rule. In the end, we will have H points for each rule. Each of the points is associated with a slope θ_j, which gives different credit to each objective. The global Q-functions will be as follows,

$$Q_{\theta_j}^*(x_n) = \sum_{l=1}^{M} \Phi^l(x_n) G_{l\theta_j}^*, \quad j = 1, \cdots, H. \tag{22}$$

In (22), the term $Q_{\theta_j}^*(x_n)$, is the global Q-function for the non-dominated Q-value, given x_n as the input. Since $G_{l\theta_j}^*$ is a (1×2) vector, $Q_{\theta_j}^*(x_n)$ will be a (1×2) vector. There is one $Q_{\theta_j}^*$ for each dashed line.

It should be noted that the MOFQL algorithm addresses the continuous action-state domain. Thus, the real Pareto front is not discrete but a continuous manifold. However,

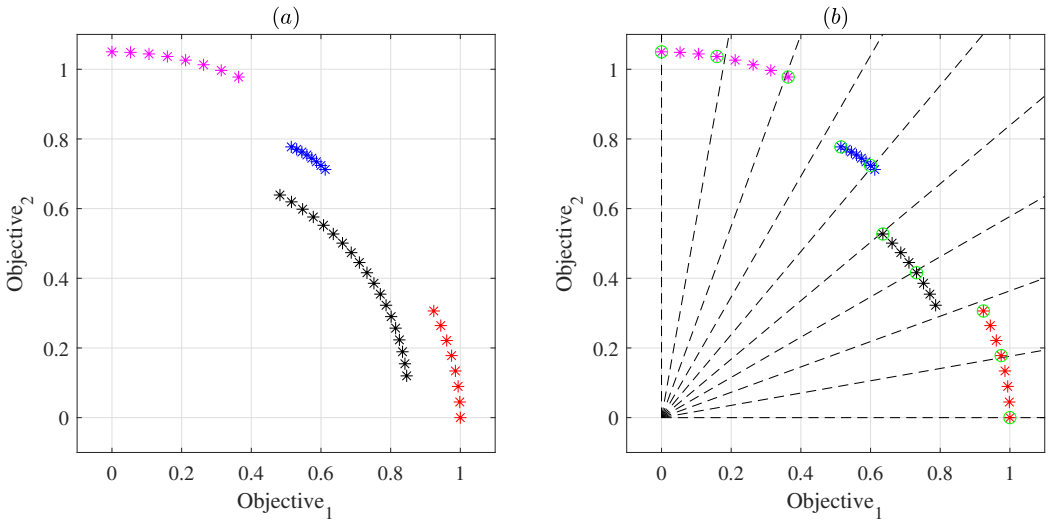

Figure 5: Golbal Q-function

the MOFQL algorithm can approximate the continuous Pareto front with discrete points. Increasing the number of calculated $Q^*_{\theta_j}$ increases the Pareto front approximation accuracy. At the same time, increasing the number of H, increases the computation effort. To control the computational complexity, we limit the number of calculated $Q^*_{\theta_j}$. We will discuss this further in section 4.4.

4.4 Updating the rules

In the single-objective FQL algorithm, we could write the temporal difference as (9). At each time step, the fired rule is updated by having the scalar reward signal and Q^*_t in (10). However, since there is only one Q^*_t, the rules are updated once in a time step. In a multi-objective FQL, we have defined several global Q-functions based on the influence of each objective as in section 4.5. Thus, we will have several optimal expected future discounted rewards. We update the fired rules for each $Q^*_{\theta_j}$ separately. Then, we combine all the updated rules in an array. Finally, we only store the non-dominated updated Q-values and eliminate the dominated Q-values. In the MOFQL algorithm, we have H different temporal differences, representing the dash lines in Figure 5 (b).

Before continuing the rest of the algorithm, let us define two mathematical operators, $\oplus$ and $\ominus$.

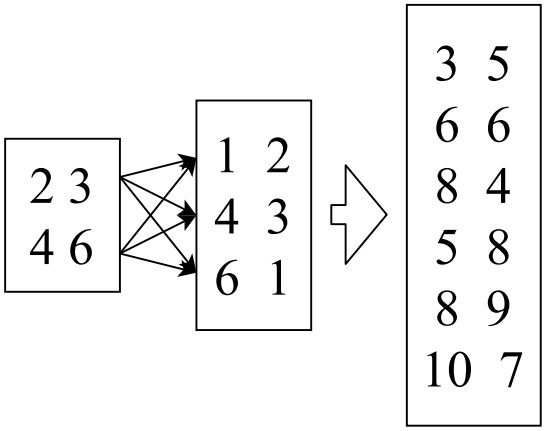

Figure 6: The $\oplus$ operator

The operator $\oplus$ and $\ominus$ are defined as,

$$\begin{aligned} V_{m\times n} \oplus W_{h\times n} &= \bigcup_{i=1}^{m}\bigcup_{j=1}^{h} V_{i(1:n)} + W_{j(1:n)} \\ V_{m\times n} \ominus W_{h\times n} &= \bigcup_{i=1}^{m}\bigcup_{j=1}^{h} V_{i(1:n)} - W_{j(1:n)}, \end{aligned} \tag{23}$$

where, $V_{i(1:n)}$ and $W_{j(1:n)}$ are the elements of ith and jth row of matrix V and W, respectively.
Example 3: The result of $V \oplus W$ for two given matrices V and W are as follows,

$$V \oplus W = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \oplus \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ 6 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 6 & 6 \\ 8 & 4 \\ 5 & 8 \\ 8 & 9 \\ 10 & 7 \end{bmatrix}. \tag{24}$$

Figure 6 illustrates the $\oplus$ operator.
The temporal difference is defined as follows,

$$\tilde{\varepsilon}_l^{\theta_j}(t+1) = \vec{r}_{t+1} + \gamma Q^*_{\theta_j}(x_{t+1}) \ominus \boldsymbol{q}_t(l, a^l), \tag{25}$$

where $\tilde{\varepsilon}_l^{\theta_j}(t+1)$ is the temporal difference with respect to the jth $Q^*_{\theta_j}(x_{t+1})$ given rule l. In (25), $\vec{r}_{t+1}$ and $Q^*_{\theta_j}(x_{t+1})$ are 1×2 vectors. However, $\boldsymbol{q}_t(l, a^l)$ is an $n \times 2$ matrix. Thus, we

use the $\ominus$ operator, which adds the 1×2 vector to each row of $\boldsymbol{q}_t(l, a^l)$, individually. The dimension of $\tilde{\varepsilon}_l^{\theta_j}(t+1)$ will be the same as $\boldsymbol{q}(l, a^l)$. Finally, $\boldsymbol{q}(l, a)$ is updated as follows.

$$\boldsymbol{q}_{t+1}(l, a_t^l) = \text{ND}\Big(\bigcup_{j=1,2,\cdots,H} (\boldsymbol{q}_t(l, a_t^l) \oplus \alpha \cdot \tilde{\varepsilon}_l^{\theta_j}(t+1) \cdot \Phi_t^l)\Big), \quad (l = 1, 2, \cdots, M). \tag{26}$$

In (26), α is the learning rate. The term Φ_t^l is the firing strength of rule l at time step t and is calculated with (3). The term H, is the number of $Q_{\theta_j}^*$s, or dash lines in Figure 5 (b), which is set by the user. The term M, is the number of rules. For each $Q_{\theta_j}^*$, the temporal difference is calculated by (25). The temporal difference is multiplied by the firing strength of the rule and the learning rate. Then, the temporal difference is added to the Q-values by $\oplus$ operator. The process is done for all H global-Q-functions individually. After taking the union of all newly calculated Q-values, the non-dominated sorting operator is affected. The new dimension of $\boldsymbol{q}_{t+1}(l, a_t^l)$ will be unknown, but it is usually more than the dimension of $\boldsymbol{q}_t(l, a_t^l)$. Since in each time step a union of multiple temporal differences is being assigned to $\boldsymbol{q}_{t+1}(l, a_t^l)$, the dimension of Q-values may increase exponentially. Thus, only a few Q-values are being kept. We keep the Q-values with the largest crowding distance and eliminate the others. Thus, we can control the Q-value density and keep them more uniform.

In order to sum up the MOFQL algorithm, the procedure is given in Algorithm 1.

Algorithm 1: MOFQL algorithm (Bi-objective)

Set H as the number of Q^*s
Initialize $q(l, a) = [0, 0]$
For Each time step **do**
  Calculate the action selection probability for all actions given each rule by (18),
  Choose an action for each rule by their selection probability,
  Compute the control signal via (2),
  Take the action and receive $\vec{r}_{t+1}$ ($\vec{r}$ is a 1×2 vector)
  Compute G_l^* via (20)
  Compute $Q_{\theta_j}^*(x_{t+1})$ via (22)
  Compute $\tilde{\varepsilon}^{\theta_j}$ for each $j = 1, \cdots, K$ with (25),
  Update $\boldsymbol{q}(l, a^l)$ with (26),
  Keep the non-dominated Q-values and delete the rest ($\boldsymbol{q}(l, a^l) \leftarrow \text{ND}(\boldsymbol{q}(l, a^l))$).
End For

4.5 Global Policy

The method discussed above finds the non-dominated Q-values as the Pareto front for each rule individually. It is beneficial to look at each rule individually for action selection and updating procedure. However, in the end, the global policies must be extracted. The proposed

MOFQL algorithm is a multi-policy learning algorithm, which means it can find more than one policy in a single implementation. To extract the global policies, we took an approach similar to [30]. In our approach, Q-values with the same influence for different objectives (similar θs) are selected from each rule. The action corresponding to each Q-value will be chosen as the output parameter for that rule. Figure 7 shows the selected Q-values for three rules. As shown in Figure 7, several lines with different slopes are depicted. Each line represents the importance of one objective with respect to the other one, as shown in (21). For instance, the solid line has the slope of 30°. The action corresponding to the closest solution of each rule, which is the closest solution to the solid line, is selected as the FLC output parameters. By doing this procedure for lines with different slopes, several policies are retrieved. The difference between extracted policies are the influence of each objective.

There are multiple benefits from the proposed algorithm. The MOFQL algorithm is able to find the non-convex regions of the Pareto front. The algorithm can estimate a continuous Pareto front manifold with discrete points. In addition, the proposed algorithm can find several policies in a single implementation without knowing the preferences of each objective.

This paper does not include a formal convergence proof for our proposed MOFQL algorithm. However, updating the rules in this paper is based on the classical FQL algorithm. Since both FQL and MOFQL algorithms follow the updating process as Q-learning, the proof precedure can follow the same way [29, 32].

5 Simulation And Discussion

We implement the proposed algorithm in section 4 to solve the reach-avoid problem presented in section 3.2. We used the hypervolume method for the action selection mechanism as presented in section 4.2. This sectioncompares the effects of choosing different hyperparameters in our algorithm. The comparisons involve the effect of the discount factor (γ), the softmax function temperature (τ), the maximum number of Q-values for each action, and the number of global Q-functions used for updating the rules (H).

5.1 Preliminaries

The reach-avoid game is presented in section 3.2. The game has one moving agent, modelled as a robot. The robot's speed is set to 5.0 $units/sec$. The game field is a square with a side of 50.0 $units$. The goal is located at (40,40), and the pit is located at (30,10). The agent's information structure consists of three inputs as shown in (12). A trained fuzzy logic controller (FLC) returns the steering angle of the robot. The axle length of the robot is set to 0.3 $units$. The FLC has five triangular membership functions for each input. Figure 8 illustrates the membership functions.

We use the hypervolume of the global policy as the performance indicator. It should be noted that the use of the hypervolume in this comparison is different from the exploration-exploitation mechanism discussed in section 4.2. The hypervolume method discussed in

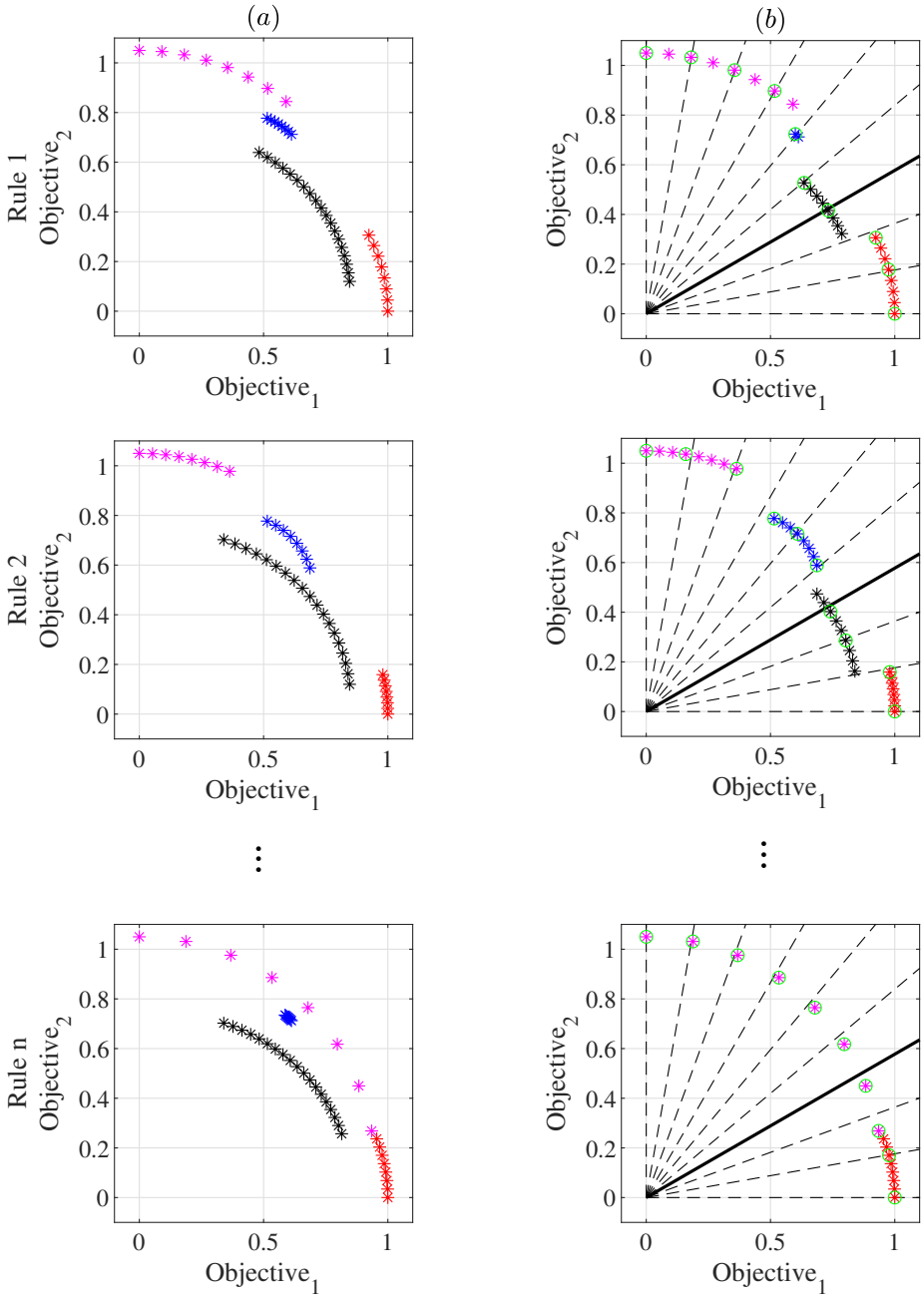

Figure 7: Action selection for a few rules

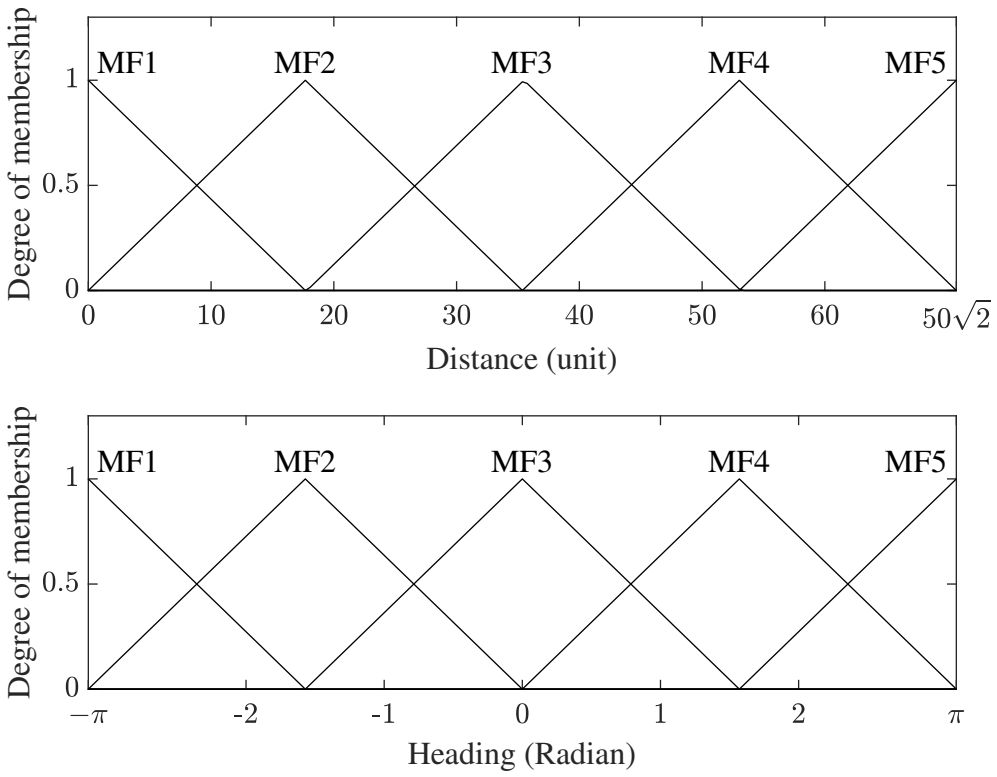

Figure 8: The membership functions for distance and angle inputs

section 4.2 is applied to each rule individually to assess each action's likelihood of being selected. However, to evaluate the algorithm performance, first, we extract the global policies as discussed in section 4.5. The game is played for each policy, and the accumulated discounted reward is calculated. The summation of the accumulated discounted reward for the global policies forms a Pareto front, and its hypervolume is reported as the performance metric. Since our objective is to maximize the accumulated reward, the higher hypervolume means a better solution.

For all simulations, the maximum epoch is set to 2,000 epochs. We observed that in less than 2,000 epochs, the hypervolume of the global policy has converged. The learning rate, α, is the same in all simulations. The initial value of α is 0.01 and it decays to 0.0001 in the final epoch. The number of Q-values for each action may increase during learning and become intractable [29]. The reason is the union operator in (26). In each time step, the union of several other non-dominated Q-values is substituted as the Q-value of the next time step. We set a limit for the maximum number of Q-values for each action given each rule. The maximum number of Q-values for each action given each rule is set to 30 Q-values that are selected with the crowding distance method [33]. The Q-values with a higher crowding distance are selected, and the rest are eliminated. We will show the effect of this selection in discussion section. In order to investigate the effect of the number of calculated global Q-functions (H), we set H to 3, 5, 7, and 9. The global Q-functions are calculated as presented in (22). It must be mentioned that before extracting the policies, the Q-values are mapped into the inteval of [0,1].

We also investigate the effect of the discount factor γ. We set the discount factor to 0.0, 0.2, 0.4, 0.6, and 0.8. A discount factor of 0.0, means that we optimize the solution to get a maximum reward on the very next step. Whereas a discount factor of 0.8 will choose a solution to maximize the reward from approximately the next five steps. The softmax temperature is set to 0, 2, 4, 6, 8, and 10. The temperature of zero means the actions are selected randomly, while the temperature of 10 means selecting the greedy action more often. In order to have a fair comparison, the reference point of the global policies' hypervolume is the same in all the simulations of this paper. The reference point is the minimum of each objective in the global policy, among all the simulations in this paper.

5.2 Simulation Results

We simulated the learning process four times for each case with different discount factors, the softmax function temperature, and the number of calculated global Q-functions. The random number generator's seed was differentfor each simulation given each case. Thus, these simulations are conducted with the same hyper-parameters and different initial random numbers for the action selection mechanism.

Let us depict the final Q-values calculation of some of the rules in a simulation in Figure 9. The first column of Figure 9 shows the union of all the action Q-values for a particular rule. The second column shows the non-dominant union of the action Q-values for the same rule. For each rule, the closest solution to the dashed lines is selected and used as the output parameter of that rule as one of the global policies. Five global policies can be created with

the five dashed lines in Figure 9. In Figure 9, $\gamma = 0.6$ and $\tau = 2.0$. The same process of retrieving the global policies is done for all other discount factors, the softmax function temperature, and the number of calculated global Q-functions.

Ten different policies are extracted in the final epoch of the training as discussed in section 4.5. Each solution represents a series of output parameters with different preferences over the reward functions. The solution with the maximum value for the first reward function represents single-objective learning with only reward 1 as its reward function. For instance, in the simulated game, we expect that the agent goes directly toward the goal if the policy corresponding to $\theta = 0$ is selected. The solution with the highest value for reward 2 represents a series of output parameters that maximizes reward 2, without considering reward 1. However, since the terminal state of the game is when the agent reaches the target, reward 1 is more important. Thus, we retrieved the Pareto fronts from $\theta = 0$ to $\theta = 45°$. Pareto optimal solutions corresponding to these regions guarantee that the agent will reach the goal. Figure 10 shows the Pareto solutions for different softmax function temperatures and different discount factors. All solutions in Figure 10 are a subset of the whole Pareto front that covers $\theta = 0$ to $\theta = 45°$. In Figure 10, the parameter H is set to 9, which means nine global Q-functions are calculated to update the output parameters.

All the cases in Figure 10 show that the solutions are non-dominated. However, it should be noted that some solutions may become dominated, especially in initial epochs, where the Q-values are not converged. In order to have a quantative comparison, we calculated the hypervolume of the Pareto fronts in Figure 10 and reported them in Table 2. The reference point for hypervolumes is set to [-0.00341,-0.228], which is 0.01 less than the minimum accumulated rewards of all the simulations in this paper. The first element of the reference point is close to zero, meaning the first argument must be a value that makes the agent always seek the goal. In Table 2, it is shown that the maximum hypervolume is for $\tau = 2.0$. However, as the discount factor increases, the accumulated reward gets larger.

Table 2: The hypervolume of the global policy for Figure 10

	$\tau = 0.0$	$\tau = 2.0$	$\tau = 4.0$	$\tau = 10.0$
$\gamma = 0.0$	0.0899	0.0984	0.1003	0.0994
$\gamma = 0.2$	0.1085	0.1201	0.1158	0.1068
$\gamma = 04$	0.1347	0.1564	0.1528	0.1260
$\gamma = 0.6$	0.1758	0.2174	0.2000	0.1783
$\gamma = 0.8$	0.2687	0.4067	0.3640	0.3022

5.2.1 Discussion

We investigate the effect of changing the hyper-parameters. We set the maximum number of each Q-value to be 30. The maximum number of each Q-value must be limited; otherwise, the number of Q-values for each action, given each rule, increases exponentially and will be intractable [29]. Choosing a very high limit for the number of the Q-values does not

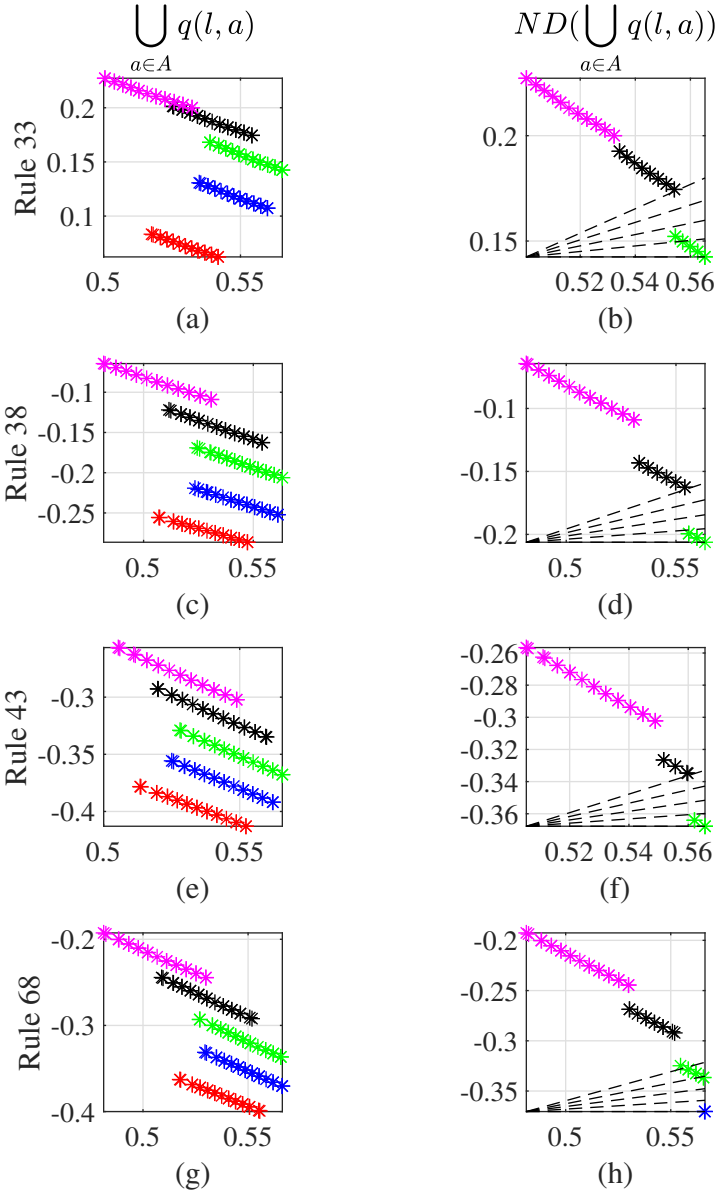

Figure 9: The final Q-values. In this simulation the hypervolume method is used as the action selector with $\tau = 2.0$. The discounted factor is 0.6, and the parameter $H = 30$.

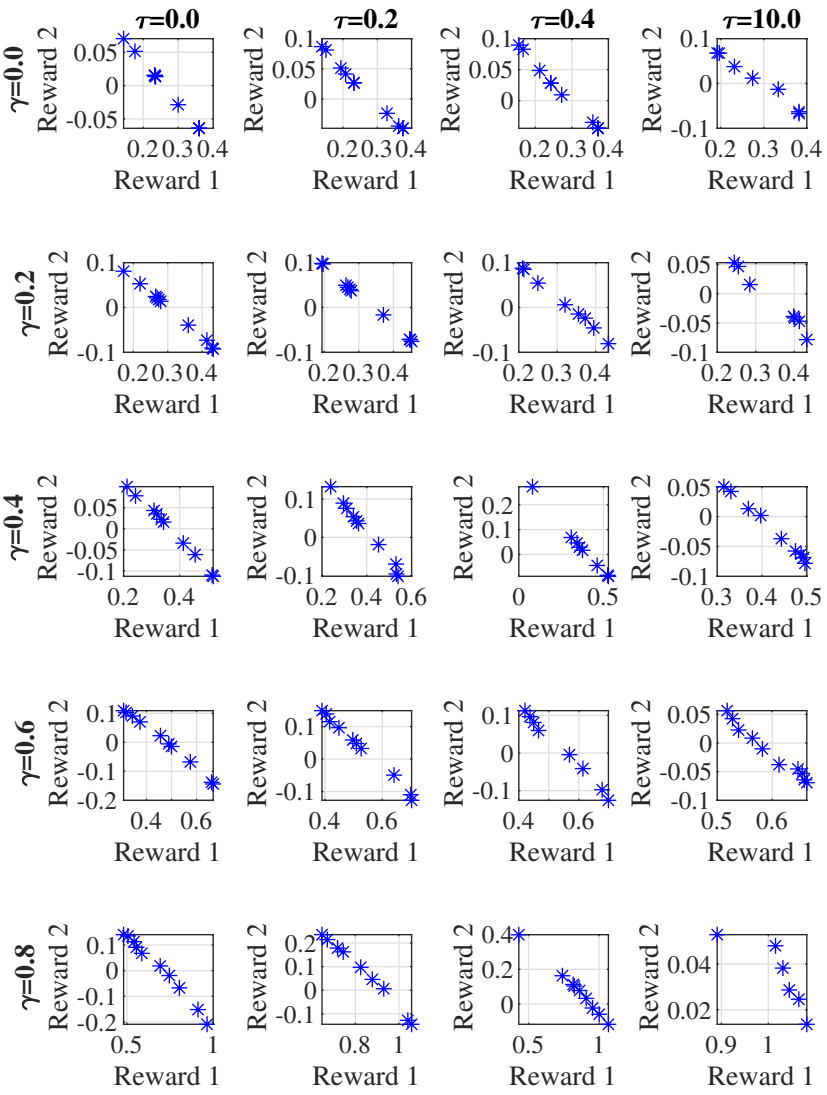

Figure 10: The Pareto optimal solutions of the game with different softmax tempratures and different discount factors for $H = 9$.

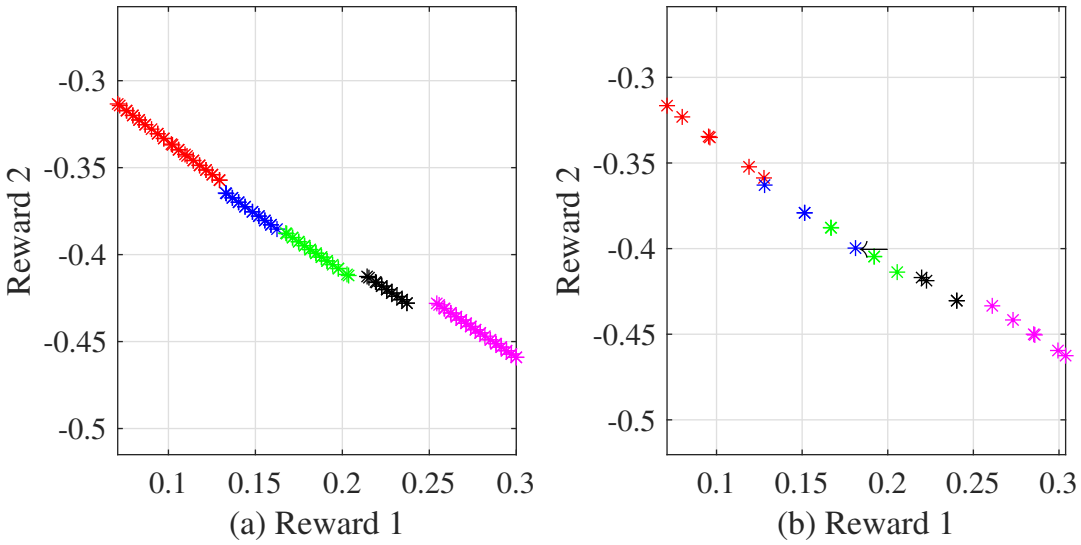

Figure 11: The effect of the limit on the maximum members of the Pareto front

necessarily increase the algorithm's accuracy. Instead, it slows down the algorithm. However, decreasing the limit jeopardizes the quality of action selection. We show this problem with an example. Figure 11 shows the non-dominant Q-values for rule 44 for two different simulations. Figure 11 (a) shows a case where the maximum number of the Q-values of each action is set to be 30. Figure 11 (b) shows a case where the maximum number of the Q-values of each action is set to be 5. As shown in Figure 11 (a), each action is fully separated from other actions. There is no Q-value of action from a colour inside the region of another colour. Figure 11 (b) shows that each action's Q-values are less dense and farther from each other. The values of the Q-values are almost the same. Thus, the limit does not have a considerable effect on the values. However, there is one Q-value of action blue in the region of action green. The action is shown with a small arrow.

Figure 12 compares the effect of the parameter H, the discount factor, and different softmax temperatures. Figure 12 (a) shows that for $\gamma = 0.0$, the number of calculated global Q-function does not affect the performance. For all Hs, the solutions are the same. The performance is also independent of the softmax temperaturesince there is no selectable peak in the figure. Although, in this case, the union of Q-values of different actions may form a Pareto front given each rule, there is only one Q-value for each action.

Figure 12 (b)-(e) show the global policy's hypervolume of cases that $\gamma > 0$. There is a peak for the softmax function temperaturein all these cases. The peak always happens for $\tau = 2.0$. Since we normalized the softmax function's input via (19), selecting $\tau = 2.0$ is also a suitable choice for different games. Figure 12 (b)-(e) also show that H does not necessarily enhance the algorithm's performance. However, in a game with more actions, the effect of H may be more visible.

We use the best performance achieved for each case, given each discount factor and the

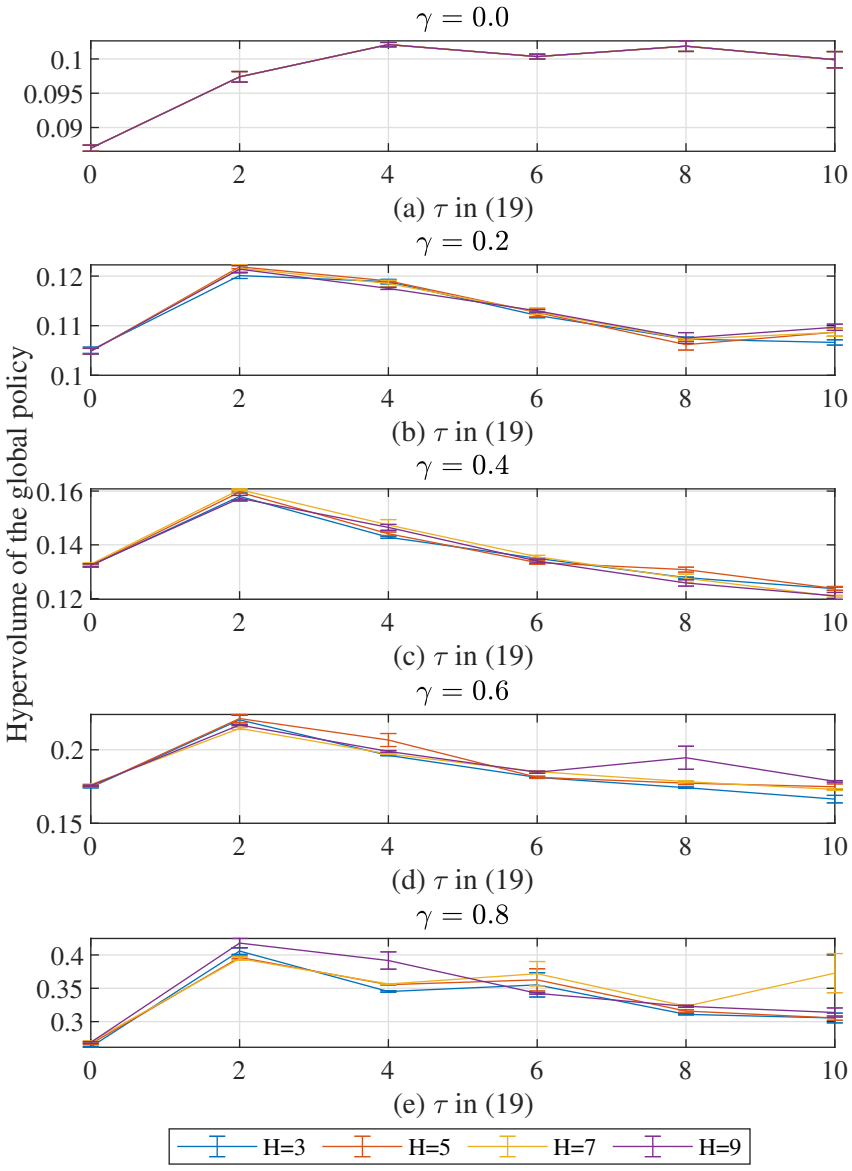

Figure 12: The hypervolume of the global policy for different softmax temperatures and action-selection mechanisms

parameter H. Figure 13 shows the convergence of the hypervolume method during the training. As we expected, the hypervolume of the global policy converged after half of the maximum epoch. For this simulation, the global policy is extracted every 50 epochs, and the hypervolume of the accumulated discounted reward is calculated. Again the reference point is assumed to be [-0.00341,-0.228], which is 0.01 less than the minimum accumulated reward. It should be mentioned that the figure is drawn for four simulations with different seeds for their random number generator. As could be expected, by increasing γ, the hypervolume increases. The reason is that the agent can see more future rewards.

Figure 14 shows the trajectory of different selected policies from the Pareto front, given different discounted factors and the parameter H. It should be mentioned that five non-dominated global policies are extracted to depict Figure 14. Five Q-values from the Pareto front of each rule are selected. The selected normalized Q-values are closest to lines with a slope of 0, 11.25, 22.50, 33.75, and 45 degrees regarding the x-axis and a reference point. The reference point has the minimum value of each dimension. It is observed that as γ increases, the trajectory will have a slightly steeper curve. However, since the reward signal is instantaneous, the difference in the trajectory is not vivid. The steeper curve is more obvious where γ is set to 0.6 and 0.8 in comparison to the smaller discounted factors. In the game of our paper, which has five actions in the action set, the effect of H is not visible.

Finally, Table 3 shows the training CPU time for different cases. We used Matlab 2019b on a Linux Ubuntu 18.04 machine to conduct the training process. The simulation time is the average of four simulations for each case. It is observed that for $\gamma = 0.0$ the simulation time is lower than other cases. On the other hand, as H increases, the calculation burden increases, and as a result, the cases with higher H have higher simulation time. It is observed that in cases that $\tau = 0.0$, the simulation time is the highest. In these cases, the actions are selected randomly, and as a result, the learning is not optimal.

6 Conclusion and future works

In this paper, a novel multi-objective reinforcement learning algorithm is proposed. The proposed algorithm is an extension of the classical FQL algorithm for the multi-objective case. A reach-avoid game is implemented as the simulation platform of this study. We used the well-known hypervolume based action selector to address the exploration-exploitation mechanism. In this study, we investigated the effect of the softmax temperature on the proposed algorithm. It was shown that the softmax temperature significantly affects the algorithm's performance. In addition, we found the best temperature that returns the highest reward. The proposed MOFQL algorithm can find the non-convex regions of the Pareto front, and it is a good choice for the problems in the continuous action-state domain.

The proposed algorithm opens a window to many applications in control systems and multi-agent systems. The MOFQL algorithm is one of the first multi-objective reinforcement learning algorithms in the continuous action-state domain. Our next study will use the proposed algorithm for multi-agent game training agents such as in [21, 34]. Our focus will be on pursuit-evasion games, where the agents' reward functions are a summation of different

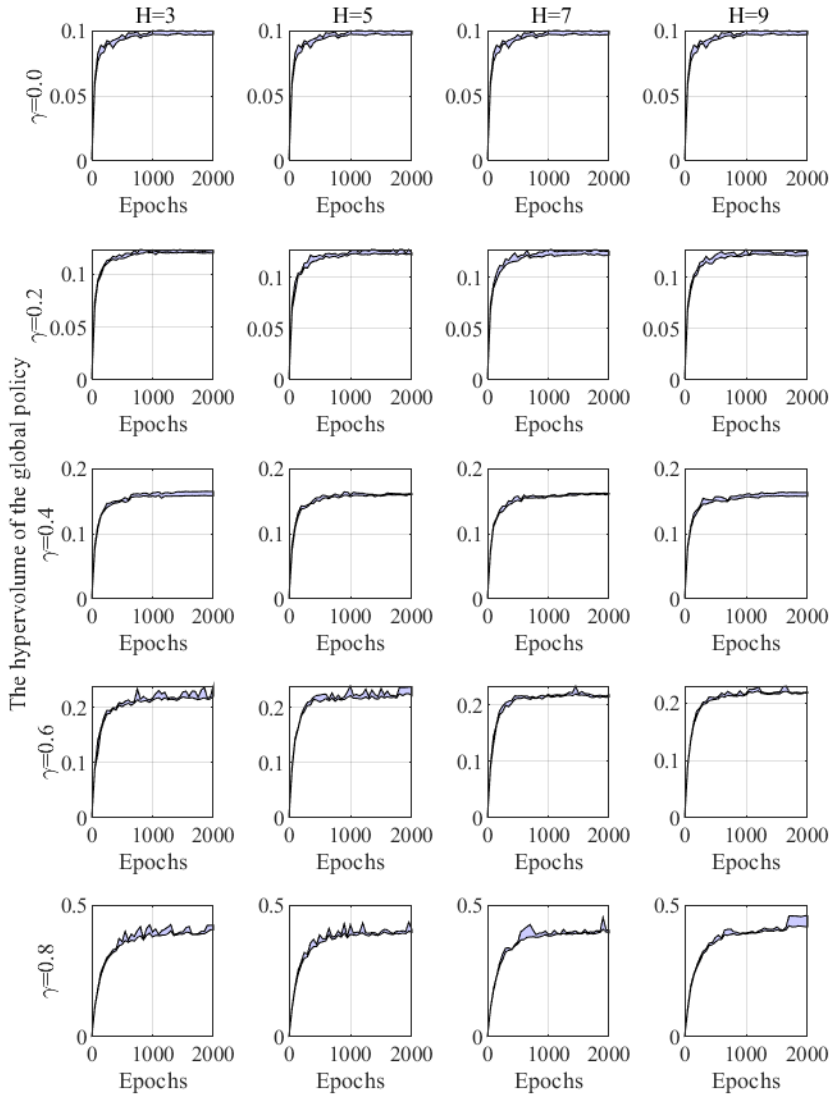

Figure 13: The hypervolume of the global policy for different action-selection mechanisms (Best τs are selected for each method given each γ)

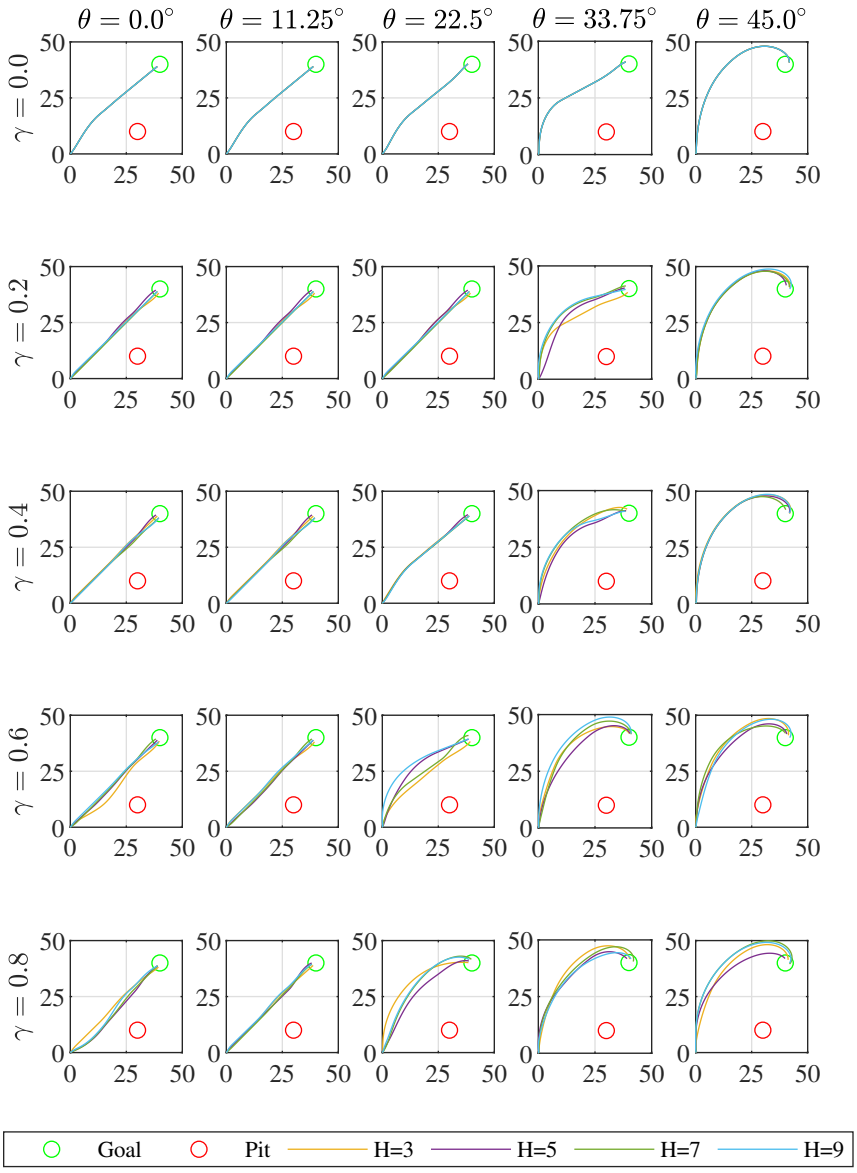

Figure 14: The agent's trajectory for different policies from the Pareto front given different γs

Table 3: The simulation time for different cases

$M=3$	$\tau=0.0$	$\tau=2.0$	$\tau=4.0$	$\tau=6.0$	$\tau=8.0$	$\tau=10.0$
$\gamma=0.0$	1580	1358	1504	1388	1383	1443
$\gamma=0.2$	17391	15719	13345	15202	12633	10811
$\gamma=0.4$	17959	13363	12428	17696	13719	10303
$\gamma=0.6$	17327	16453	15031	11811	12770	12250
$\gamma=0.8$	16528	14160	13595	11248	14689	11817

$M=5$	$\tau=0.0$	$\tau=2.0$	$\tau=4.0$	$\tau=6.0$	$\tau=8.0$	$\tau=10.0$
$\gamma=0.0$	1965	1514	1644	1543	1592	1624
$\gamma=0.2$	22128	18050	17894	17976	18562	11407
$\gamma=0.4$	22412	23044	19470	17740	13984	13429
$\gamma=0.6$	21052	19887	18480	20157	14839	13618
$\gamma=0.8$	21019	20961	13374	16614	11436	11196

$M=7$	$\tau=0.0$	$\tau=2.0$	$\tau=4.0$	$\tau=6.0$	$\tau=8.0$	$\tau=10.0$
$\gamma=0.0$	1879	1606	1748	1634	1685	1752
$\gamma=0.2$	28062	24133	25765	24247	19799	15501
$\gamma=0.4$	29888	22214	27974	24045	23394	20888
$\gamma=0.6$	27955	27325	18042	23184	18802	17587
$\gamma=0.8$	25410	23153	22658	20599	21455	19670

$M=9$	$\tau=0.0$	$\tau=2.0$	$\tau=4.0$	$\tau=6.0$	$\tau=8.0$	$\tau=10.0$
$\gamma=0.0$	2017	1731	1882	1768	1791	1875
$\gamma=0.2$	35066	29782	30112	31786	27924	21485
$\gamma=0.4$	34243	29247	34956	28689	24655	27818
$\gamma=0.6$	33780	33075	30937	27038	23092	21229
$\gamma=0.8$	34694	31990	21368	22656	20814	20683

conflicting reward signals. One interesting question in studying a multi-agent multi-objective game will be how the Pareto front will look. In addition, we will study a new multi-objective reinforcement algorithm based on the fuzzy actor-critic learning algorithm.

Acknowledgement

This research is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). (No. RGPIN-2017-06379 and No. RGPIN-2017-06261)

References

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations : Theory and application to reward shaping," *Sixteenth International Conference on Machine Learning*, vol. 3, pp. 278–287, 1999.

[4] M. Babes, E. M. De Cote, and M. L. Littman, "Social reward shaping in the Prisoner's dilemma," *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 3, no. Aamas, pp. 1357–1360, 2008.

[5] H. Zhang, D. C. Parkes, and Y. Chen, "Policy teaching through reward function learning," *Proceedings of the ACM Conference on Electronic Commerce*, pp. 295–304, 2009.

[6] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[7] D. J. White, "The set of efficient solutions for multiple objective shortest path problems," *Computers and Operations Research*, vol. 9, no. 2, pp. 101–107, 1982.

[8] A. Castelletti, G. Corani, A. E. Rizzoli, R. Soncini Sessa, and E. Weber, "Reinforcement learning in the operational management of a water system," *Modelling and Control in Environmental Issues 2001*, pp. 325–330, 2002.

[9] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," pp. 372–378, 2008.

[10] T. Brys, A. Harutyunyan, P. Vrancx, A. Nowé, and M. E. Taylor, "Multi-objectivization and ensembles of shapings in reinforcement learning," *Neurocomputing*, vol. 263, pp. 48–59, 2017.

[11] N. Nariman-Zadeh, M. Salehpour, A. Jamali, and E. Haghgoo, "Pareto optimization of a five-degree of freedom vehicle vibration model using a multi-objective uniform-diversity genetic algorithm (MUGA)," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 543–551, 2010.

[12] I. Showalter and H. M. Schwartz, "Neuromodulated multiobjective evolutionary neurocontrollers without speciation," *Evolutionary Intelligence*, vol. 14, no. 4, pp. 1415–1430, 2021.

[13] M. A. Khamis and W. E. S. A. Gomaa, "Enhanced multiagent multi-objective reinforcement learning for urban traffic light control," *Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012*, vol. 1, no. 2, pp. 586–591, 2012.

[14] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning," vol. 14, 2002.

[15] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.

[16] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*, pp. 601–608, 2005.

[17] L. Barrett and S. Narayanan, "Learning all optimal policies with multiple criteria," *Proceedings of the 25th International Conference on Machine Learning*, pp. 41–47, 2008.

[18] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learning," in *IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 659–662, 1997.

[19] H. Daellenbach and C. De Kluyver, "Note on multiple objective dynamic programming," *Journal of the Operational Research Society*, pp. 591–594, 1980.

[20] S. Mannor and N. Shimkin, "A geometric approach to multi-criterion reinforcement learning," *Journal of Machine Learning Research*, vol. 5, pp. 325–360, 2004.

[21] A. Asgharnia, H. M. Schwartz, and M. Atia, "Deception in a multi-agent adversarial game: the game of guarding several territories," *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 1321–1327, 2020.

[22] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2011.

[23] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Hypervolume-based multi-objective reinforcement learning," pp. 352–366, 2013.

[24] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[25] M. Pirotta, S. Parisi, and M. Restelli, "Multi-objective reinforcement learning with continuous pareto frontier approximation," *Proceedings of the National Conference on Artificial Intelligence*, vol. 4, pp. 2928–2934, 2015.

[26] S. Parisi, M. Pirotta, and M. Restelli, "Multi-objective reinforcement learning through continuous pareto manifold approximation," *Journal of Artificial Intelligence Research*, vol. 57, pp. 187–227, 2016.

[27] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, "Multi-objective deep reinforcement learning," *arXiv preprint arXiv:1610.02707*, 2016.

[28] P. Vamplew, R. Dazeley, and C. Foale, "Softmax exploration strategies for multiobjective reinforcement learning," *Neurocomputing*, vol. 263, pp. 74–86, 2017.

[29] M. Ruiz-Montiel, L. Mandow, and J. L. Pérez-de-la Cruz, "A temporal difference method for multi-objective reinforcement learning," *Neurocomputing*, vol. 263, pp. 15–25, 2017.

[30] Z. Daavarani Asl, V. Derhami, and M. Yazdian-Dehkordi, "A new approach on multi-agent multi-objective reinforcement learning based on agents' preferences," *19th CSI International Symposium on Artificial Intelligence and Signal Processing, AISP 2017*, vol. 2018-Janua, pp. 75–79, 2018.

[31] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on evolutionary computation*, vol. 7, no. 2, pp. 117–132, 2003.

[32] C. Watkins and P. Dayan, "Q-Learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[34] J. Zhang, Z. Wang, and H. Zhang, "Data-based optimal control of multiagent systems: A reinforcement learning design approach," *IEEE transactions on cybernetics*, vol. 49, no. 12, pp. 4441–4449, 2018.