Dynamic Thresholded Lexicographic Ordering

Conor F. Hayes
School of Computer Science
National University of Ireland Galway
Ireland
c.hayes13@nuigalway.ie

Enda Howley
School of Computer Science
National University of Ireland Galway
Ireland
enda.howley@nuigalway.ie

Patrick Mannion
School of Computer Science
National University of Ireland Galway
Ireland
patrick.mannion@nuigalway.ie

ABSTRACT

The goal of multi-objective problems is to find solutions that balance different objectives. When solving multi-objective problems using reinforcement learning linear scalarisation techniques are generally used, however system expertise is required to optimise the weights for linear scalarisation. Thresholded Lexicographic Ordering (TLO) is one technique that avoids the need for an expert to specify weights; instead a system designer can directly specify a preferred ordering over objectives, along with a desired threshold value for each objective. In this paper we propose a novel algorithm to dynamically set thresholds for use with TLO. We also present the first evaluation of TLO in a complex multi-objective multi-agent problem, the Dynamic Economic Emissions Dispatch domain. Our empirical results demonstrate that TLO with our dynamic thresholding algorithm achieves superior results when compared with a hand-tuned linear scalarisation method from previously published work.

KEYWORDS

Multi-objective; Reinforcement Learning; Multi-agent systems

1 INTRODUCTION

The current reinforcement learning literature concentrates mainly on problems with a single objective, while most real-world problems contain multiple objectives. For example in energy generation, engineers may want to minimise both the cost and emissions produced by the energy generators. Therefore when applying reinforcement learning to real-world problems it is important to expand reinforcement learning algorithms to include multiple objectives. Multi-objective optimisation is the process of optimising multiple objectives concurrently; these objectives are conflicting. Linear scalarisation functions are widely used to transform multi-objective return vectors into a single scalar reward in reinforcement learning. Setting the required weights for linear scalarisation techniques requires system expertise and is not an intuitive process, as a small change in the weight space can lead to a large change in the outcome during execution.

In this work we will show how linear scalarisation techniques can be replaced with a non-linear action selection technique, known as Thresholded Lexicographic Ordering (TLO) [4, 11] in multi-objective multi-agent reinforcement learning (MOMARL) problems. Thresholded Lexicographic Ordering reduces complexity in multi-objective optimisation problems by removing the weight selection process entirely, instead enabling a user to specify their preferences for possible outcomes by setting a lexicographic order over objectives, along with target thresholds for each objective. We introduce a novel method for dynamically setting thresholds that can completely remove the requirement for previous system knowledge when setting thresholds manually. We also provide the first experimental evaluation of TLO in a multi-objective multi-agent setting, the Dynamic Economic Emissions Dispatch problem. Our empirical results explore the effect of objective ordering on system performance, and demonstrate that our dynamic TLO method achieves superior performance when compared to a previously published expert-tuned linear scalarisation approach.

2 RELATED WORK

2.1 Reinforcement Learning

Reinforcement learning (RL) is a machine learning pardigram where autonomous agents learn how to complete a defined task through experience. An agent receives a scalar reward from the environment for previously taken actions and its goal is to maximise its own scalar reward. Markov Decision Processes are considered the standard when defining problems for single-agent RL problems [14]. A MDP consists of a set of states, a set of actions, a reward function and a transition function, i.e. a tuple $< S, A, T, R >$. The reward function R defines a scalar reward, r given to an agent for performing an action in a state. When an agent is in state s, selecting action a, the agent will transition to state s'. The probability of transition to state s', having taken action a while in state s is denoted by $T(s, a, s')$ and gives a reward $r = R(s, a, s')$. An agent acts in an environment based on its policy π. The goal of an MDP is to find the best policy for an agent in an environment. In order for an agent to learn an optimal policy, π^*, an agent must learn about all states in a given environment. To find an optimal policy an agent must find a balance between exploration (exploring unknown states in an environment) and exploitation (exploiting what the agent has already learned about the environment). One strategy that strikes a balance between exploration and exploitation is the ϵ - greedy algorithm.

RL can be classified into two paradigms: model-based (e.g Dyna, Rmax) and model-free (e.g Q-learning, SARSA). Model-based approaches attempt to learn the transition function T. While, in contrast, model-free approaches do not require any knowledge of the transition function T. Model-free RL provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions without requiring agents to build maps of the domains [13]. Therefore, model-free learners sample the underlying MDP directly in the form of value function estimates (Q-values). Each Q-value represents the expected reward for each state-action pair. This aids the agent in selecting what action to take when in a given state. The agent generally takes the

action with the highest Q-value out of the available actions when acting greedily.

Q-learning [13] is a commonly used RL algorithm. Q-learning is a model-free algorithm that has been shown to converge to the optimum policy with probability 1 in discrete environments, given sufficient experience of all state-action pair. Q-learning updates the Q-values using the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)] \quad (1)$$

where α is the learning rate and γ is the discount factor.

2.2 Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning (MARL) is an expansion on traditional single-agent RL, where multiple autonomous agents learn in an environment. In MARL the single-agent MDP framework becomes inadequate and therefore we must generalise the MDP to Stochastic Games (SGs). A SG is defined as a tuple $< S, A_{1\ldots n}, T, R_{1\ldots n} >$, where n is the number of agents, S is the set of states, A_i is the set of actions for agent i, T is the transition function, and R_i is the reward for agent i. A SG can be thought of as an extension to an MDP. An MDP is a SG where the number of agents is equal to 1. The next state of the system depends on the joint actions of all the agents in the SG. SGs can be fully co-operative, fully competitive or mixed. Multiple individual learners can be deployed to learn a SG. In this case, multiple agents individually learn using a single-agent RL algorithm (e.g Q-learning).

2.3 Multi-Objective Reinforcement Learning

A distinction between single-objective and multi-objective reinforcement learning is how the reward is structured. In single agent reinforcement learning, the reward is a scalar. To expand reinforcement learning to be used in the multi-objective domain we must change the reward function, R, so that it returns vectorial rewards rather than scalars. The reward function, **R**, for multi-objective reinforcement learning therefore returns a vector **r** that has a value component per objective.

2.4 Multi-Agent Credit Assignment Structures

How an agent learns to act in an environment is guided by its reward signal. Agents in cooperative MAS can receive the same reward or different rewards depending on the credit assignment structure used.

Previous work by Yliniemi et al. [16] identified the importance of using appropriate credit assignment structures in multi-objective MARL problem domains. Experimental results presented by Yliniemi et al. [16] demonstrated that difference rewards are a very promising approach for learning good-joint policies in multi-objective MARL problems.

The **global reward** (G) provides feedback to the agents which is based on the utility of the entire system. The global reward encourages agents to act in a way that benefits the overall system. But the contribution of each agent's actions to the performance of the system is not well defined (i.e. the global reward is "noisy"). In a system where agents learn using the global reward all agents receive the same reward. This can lead to agents receiving a positive reward for actions that are not in the best interest of the system and vice versa.

The **difference reward** (D_i) is a shaped reward signal that aims to quantify each agent's contribution to the system performance [15]. D is calculated by subtracting the global performance for a theoretical system without the contribution of agent i from the true global performance. This can be defined as follows:

$$D_i(z) = G(z) - G(z_{-i}) \quad (2)$$

where $G(z)$ is the global system utility, $G(z_{-i})$ is the global utility for a theoretical system without the contribution of agent i, and $D_i(z)$ is the difference reward given to agent i.

2.5 Thresholded Lexicographic Ordering

Thresholded Lexicographic Ordering (TLO), first introduced by Gabor et al [4], allows an agent in a multi-objective environment to select actions which prioritise objectives in lexicographic order, subject to desired minimum thresholds for performance on each objective.

Vamplew et al. [11], applied TLO with Q-learning (TLQ-learning) in a multi-objective single-agent environment. The experimental results showed that TLQ-learning converged to a Pareto optimal solution in fewer episodes when compared to scalarised Q-learning in a multi-objective single-agent problem domain.

TLQ-learning utilises a different action selection algorithm compared to traditional Q-learning. This is outlined in Algorithm 1:

- Let n denote the number of objectives, labeled from 1..n
- Let A denote the set of available actions
- Let C_j be the threshold value (minimum acceptable value) for objective j, as defined by the constraint for that objective (note: objective n will be unconstrained, hence $C_n = +\infty$)

$$CQ_{s,a,j} \leftarrow min(Q_{s,a,j}, C_j) \quad (3)$$

In state s, the greedy action a' is selected such that:

$$superior(CQ_{s,a'}, CQ_{s,a}, 1) == True \; \forall a \in A \quad (4)$$

where superior($CQ_{s,a'}$, $CQ_{s,a}, i$) is recursively defined in Algorithm 1 [11]:

Algorithm 1: TLQ-learning Action Selection

```
if CQ_{s,a',i} > CQ_{s,a,i} then
    return true;
end
if CQ_{s,a',i} = CQ_{s,a,i} then
    if i = n then
        return true;
    else
        return superior(CQ_{s,a'} , CQ_{s,a}, i + 1);
    end
else
    return false;
end
```

2.6 Ordering of Objectives

Vamplew et al. [11] evaluated the effect of the ordering of optimisation of objectives on system performance in single-agent MORL domains. Ordering of objectives is subjective; $user_a$ may have a preference for optimising $objective_1$ first, while $user_b$ may have a preference to optimise $objective_2$.

In TLO the user must state their objective preferential order before running a system, this is a limitation as when problems scale to a large number of objectives, users might not be aware of their exact preferences. Recently, it was demonstrated by Zintgraf et al. [17] that it is possible to model expert users' preferences over objectives using a utility function; however this work required a lengthy preference collection phase, and these preference-based utility functions have not yet been evaluated empirically as scalarisation functions for reinforcement learning in MORL domains. Our dynamic TLO method avoids the need for a lengthy preference collection phase, asking a user to specify an ordering over objectives only once before learning begins.

As previously mentioned, agents learning with Q-learning receive a scalar reward while agents learning using TLO receive a vector based reward defined as follows:

$$\mathbf{R} = [-o_1, -o_2, ..., -o_n] \tag{5}$$

where each component o represents the global reward for an objective and n is the number of objectives.

3 NOVEL TECHNIQUES FOR THRESHOLDED LEXICOGRAPHIC ORDERING

One shortcoming in current MOMARL approaches is the lack of a method to specify user preferences over objectives in an intuitive manner. In real-world multi-agent environments, objectives have varying degrees of importance, and current state-of-the-art fails to capture how important an objective is to a given user. Previous works on multi-objective multi-agent learning have used linear scalarisation techniques to specify the relative importance of objectives [8, 16]. The weights chosen in this approach can only be found through extensive parameter sweeps by expert system users. When using linear scalarisation values, a small change in the input weight value can lead to a large change in the output. It was noted by Van Moffaert et al. [12] that linear scalarisation functions are not suitable as a basis for an exploration strategy as they are biased towards particular actions, while ignoring other Pareto dominating actions. Our dynamic Thresholded Lexicographic Ordering method, which is a non-linear technique, aims to remedy these shortcomings. TLO has future potential in many multi-objective multi-agent domains, such as: natural resource management [3], traffic signal control [5], scheduling tasks on multi-core processors [6] and ground water monitoring [1].

3.1 Multi-Agent Thresholding

The current literature has only studied TLO with regard to multi-objective single-agent RL. In this paper we present, for the first time, a detailed study of TLO with MOMARL.

We can consider a multi-agent environment to be stochastic; from an individual agent's perspective, the environment is stochastic as the actions of each agent affect the global reward. This is also the first time TLO has been evaluated empirically in a stochastic environment.

To expand TLO beyond a single agent setting we must also reconsider how thresholds are defined. In a single agent domain, only one agent interacts with a threshold making threshold selection an easy task. In a cooperative setting, the threshold for global system performance on each objective is shared by all agents.

Thresholds in our work are based on the global system reward. Setting a threshold based on the local reward would require a threshold per agent per timestep. This would be a complex task, given at each timestep an agent can take any allowed action. For example, in the Dynamic Economic Emissions Dispatch problem (Section 4), where 9 agents control 9 generators, it would be necessary for each agent to have its own individual threshold. Setting accurate individual thresholds in this problem domain would require a different threshold for each of the 24 timesteps and would require detailed knowledge of the effect of each agent's actions at each timestep. As we are interested in simplifying the process of specifying user preferences in MOMARL, we adopt threshold values that are set globally and are shared by all agents. Each agent will keep track of the global threshold and subtract the global reward for each timestep during learning, leaving a remaining threshold value for later timesteps. In theory by the end of an episode, the remaining threshold value will be 0 for each objective.

3.2 Threshold Methodology

The current state-of-the-art literature on TLO does not describe any methodology for setting thresholds. Several authors (e.g. [7, 11]) set thresholds based on their experience with the specific problem domain, adding further unnecessary complexity and painting threshold selection as a trial and error process rather than a science.

In TLO a threshold is specified for each objective, with the exception of the final objective. The system attempts to reach the specified threshold for each objective, while the final objective is optimised. In this paper, we have used a threshold vector $\mathbf{T}$ that contains the desired outcome for an entire episode. $\mathbf{T}$ can be defined as the following:

$$\mathbf{T} = [goalValue_1, ..., goalValue_{n-1}] \tag{6}$$

where n is the number of objectives and $goalValue$ is the desired outcome for each objective.

Fixed thresholds rely on a user having previous knowledge of an advantageous outcome. The user will specify a desired system outcome for each required objective. Fixed thresholds have the limitation of requiring specialist knowledge of the system to ensure a useful threshold is set. Refining fixed thresholds to obtain an optimal solution takes significant expertise and is a trial and error process.

To address this shortcoming, we propose **dynamic thresholds** set by the system through evaluations of the system's previous performance. Each threshold is defined by the system via feedback received on the current and previous performance. Using the information gathered via feedback a useful threshold value can be defined. Dynamic thresholds do not require any specialist knowledge of an optimal outcome for a system and therefore simplify the process of specifying user preferences for MOMARL.

When using dynamic thresholds we do not require a minimum acceptable value of each objective from the user, just the user's preference over how they order the objectives. For example, a user of an autonomous vehicle might want to get to their destination as quickly as possible without any regard for obeying traffic laws. Another user might want to get to their destination in the quickest time possible, but without breaking any laws. The importance of each objective varies significantly depending on the user, but they may not know how to quantify their preferences with exact threshold values. Thus, we have defined a methodology to incorporate the subjective importance of objectives without resorting to manually specifying weights (as in linear scalarisation) and/or thresholds (as in previous TLO approaches).

3.3 Continuous Improvement Thresholding

Continuous Improvement Thresholding (CIT) is a novel form of optimistic dynamic thresholding introduced in this paper. CIT enhances performance while learning by setting new threshold values without the need for expert user input.

At each episode the previous system performance is evaluated for each objective. If the previous performance, p_x, for an objective is the best performance in a set of previous performances, $\{p_1, p_2, ..., p_n\}$, where n is equal to the number of episodes, then the threshold, $\mathbf{T}$, for objective, o, is defined as:

$$\mathbf{T_o} = p_x \tag{7}$$

Algorithm 2: CIT Selection Algorithm

```
At episode = 1, set thresholds T = 0 for all objectives
At episode = 1, p_episode = 0 for all objectives
for episode in numEpisodes do
    for objective in numObjectives do
        add p_episode to {p_numEpisodes}
        if p_episode > p_optimal then
            p_optimal = p_episode
        end
        T_objective = p_optimal
    end
end
```

When setting a fixed threshold a user will specify an optimal value based on previous experience. When defining dynamic thresholds a trade-off between optimistic and realistic thresholds must be made. An optimistic threshold is set based on the best previous performance of the system, e.g CIT. A realistic threshold is set based on finding a balance between an optimal threshold and an obtainable threshold. When using optimistic thresholds in stochastic problem domains (e.g. domains that are noisy due to the effect of other agents learning), optimistic thresholds can become unobtainable and the system only optimises the first objective. In order to counteract this effect we propose the following enhancements that can be used with the CIT algorithm in stochastic problem domains:

Enhancement 1 applies a 10% decrease buffer to the previous best system performance for each objective and sets this as the threshold.

Enhancement 2 uses the average of the best previous performance with a fixed window (e.g. the last 200 episodes' performances) for each objective, and sets this value as the threshold.

Enhancement 3 identifies when the CIT algorithm has established an unobtainable threshold, and using this information sets a threshold based on the average of the previous values.

The proposed enhancements aim to preserve the importance of a particular objective to a user. The importance of objectives for users is lost when using optimistic thresholds. Using the CIT algorithm on its own optimises the first objective with high importance, while using CIT with any of the proposed enhancements optimises all objectives individually. CIT and the proposed enhancements attempt to capture how important an objective is to a user in MOMARL without explicit input apart from specifying the ordering of objectives.

4 THE DYNAMIC ECONOMIC EMISSIONS DISPATCH (DEED) DOMAIN

DEED is a problem first introduced by Basu [2]. In DEED a number of electrical generators must be scheduled in order to provide power for a population while minimizing both cost and emissions. This problem has received attention in recent years, and techniques such as Particle Swarm Optimization [9] and linear scalarised MOMARL [8] have been applied to it. For our experimental work, we use a multi-objective stochastic game version of the DEED domain [8].

Since this problem will use multiple agents, an agent will be used per generator. There will be an agent for each directly controlled generator. In DEED there are 10 generators but only 9 are directly controlled since the slack generator, when $n = 1$, has its values set indirectly to account for the difference between the power demand and the actual power generated by the 9 directly controlled generators. There will be 9 agents used where $i \in \{2, ..., 10\}$. Each agent will set the power output of its generator $n = i$ at every timestep m.

4.1 Reward Structures in the DEED Domain

In order to measure the system performance, we use emissions and cost functions [2, 8]. f_c^L computes the local cost for each generator, n, over hour m:

$$f_c^L(n, m) = a_n + b_n P_{nm} + c_n (P_{nm})^2 + |d_n sin\{e_n (P_n^{min} - P_{nm})\}| \tag{8}$$

Therefore the the global cost function f_c^G for all generators over hour m is:

$$f_c^G(m) = \sum_{n=1}^{N} f_c^L(n, m) \tag{9}$$

f_e^L computes the local emissions for each generator, n, over hour m. This equation is defined below:

$$f_e^L(n, m) = E(a_n + b_n P_{nm} + \gamma_n (P_{nm})^2 + \eta \exp \delta P_{nm}) \tag{10}$$

where E = 10 [8] the emissions scaling factor, chosen so the magnitude of the local emissions function f_e^L matches that of the local cost function f_c^L. It follows that the global emissions function f_e^G

for all generators over hour m is:

$$f_e^G(m) = \sum_{n=1}^{N} f_e^L(n, m) \tag{11}$$

The state for each is defined as a vector that contains the change in power demand since the previous timestep, ΔP_D, and the previous power output of the generator n, P_{nm}. Therefore the state vector for agent i (controlling generator n) at time m is:

$$s_{im} = [\Delta P_{Dm}, P_{n(m-1)}] \tag{12}$$

The power limits and the ramp limits for the slack generator must be taken into consideration. A global penalty function f_p^G based on the static penalty method developed by Smith et al. [10]. This function captures the violations of these constraints:

$$f_p^G(m) = \sum_{v=1}^{V} C(|h_v + 1|\delta_v) \tag{13}$$

It is important to note the penalty function is an additional objective that will need to be optimized in conjunction with the cost and emissions functions [8]. All equations and parameters absent from this paper that are required to implement this problem domain can be found in the works of Basu [2] and Mannion et al. [8].

4.2 Calculating Counterfactuals

The counterfactuals outlined in this section are used for calculating the difference reward [8]. The counterfactual cost, emissions and violation terms for an agent, i, are calculated by assuming that the agent did not chose a new power output value in the previous timestep. The counterfactual for cost is calculated by the following:

$$f_c^{G(z-i)} = \sum_{\substack{n=1 \\ n \neq i}}^{N} f_c^L(n, m) + f_c^L(i, m-1) \tag{14}$$

The counterfactual for emissions is calculated by the following:

$$f_e^{G(z-i)} = \sum_{\substack{n=1 \\ n \neq i}}^{N} f_e^L(n, m) + f_e^L(i, m-1) \tag{15}$$

The output of the counterfactual version of $f_p^{G(z-i)}$ of the penalty function f_p^G is calculated using Eqn. 13. The required equations for defining parameters to find $f_p^{G(z-i)}$ can be found in the work by Mannion et al. [8].

4.3 Scalarisation of Objectives

The difference and global reward structures for each objective are scalarised with linear scalarisation (+) [8]:

$$R_+ = -\sum_{o=1}^{O} w_o f_o \tag{16}$$

where w_o is the objective weight, f_o is the objective function (global or difference, where appropriate) and the generic R is replaced by G or D. The objective weight used as $w_c = 0.225$, $w_e = 0.275$ and $w_p = 0.5$. These are hand-tuned values from prior work by other authors, and will serve as a comparison to our dynamic thresholding approach [8]. The agents receive one of these scalarised reward signals while learning: $G(+)$, $D(+)$. Agents learning with TLQ-learning do not receive any scalarisation.

4.4 TLQ-Learning Reward Structure

Agents implementing TLO learn with a variant of the global reward using a vector-based structure. The ordering of objectives within the reward vector $\mathbf{r}$ (defined in Equation 5) is subject to change depending on which objective is being optimised first. For example, a system optimising objectives in the following order; violations, cost, emissions will have the following reward structure:

$$\mathbf{r} = [f_p^G(m), f_c^G(m), f_e^G(m)] \tag{17}$$

4.5 Action Selection

Each agent has a set of 101 possible actions, $A^* = \{0, 1, ..., 99, 100\}$, and each action represents a different percentage value of the operating range of the generator [8]. All agents select actions using an ϵ-greedy strategy, while agents learning with Q-learning use the standard Q-learning action selection algorithm. Agents learning with TLQ-learning use the action selection algorithm defined in Algorithm 1.

5 EXPERIMENT EVALUATION

5.1 Experimental Procedure

In this version of the DEED problem the agents learn for 20,000 episodes, where each episode contains 24 hours. The power demand profile for this experiment is the same power demand profile used in previous works by other authors [2, 8, 9]. The learning parameters for all agents are as follows: $\alpha = 0.10$, $\gamma = 1$, $\epsilon = 0$. Setting ϵ to 0 means that optimistic initial Q values (i.e. all Q values set to 0.0) are used to encourage exploration. These values were selected following parameter sweeps to determine the best performing settings.

5.2 Effect of Ordering of Objectives on System Performance

The ordering of objectives when using TLQ-learning has a significant impact on the overall system performance. For the DEED problem domain, we theorised that the violations objective should always be optimised first as in a real-world system generator power violations would not be tolerated or possible without overall system failure. Using this theory as a basis for experimentation we tested this idea by optimising emissions (TLO Violations - Emissions) and cost (TLO Violations - Cost), then tested our system by optimising cost followed by violations (TLO Cost - Violations).

We can see noticeable performance fluctuations when the order of the objectives varies. In this section we have only evaluated the effect of using dynamic thresholds. In Figure 1 when we optimise the the cost objective first (TLO Cost - Violations) the system performance suffers as the agents fail to lower the violations objective to a favorable level.

In Figure 1, when violations are optimised first, the system performance is enhanced compared to when cost is optimised first. Optimising with TLO Violation - Emissions has the greatest influence on the system for the violations objective. Optimising with

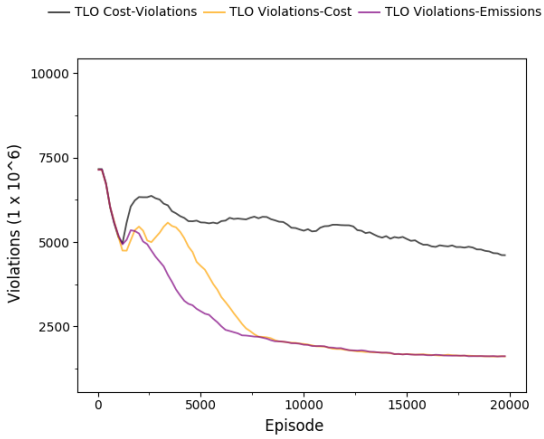

Figure 1: Effect of Ordering of Objectives for Violations

TLO Violations - Cost finds a better solution for violations, although it takes the system longer to learn this solution when compared to TLO Violations - Emissions.

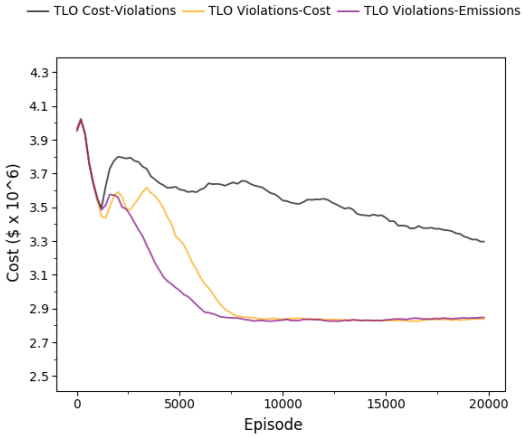

Figure 2: Effect of Ordering of Objectives for Cost

In Figure 3, TLO Cost-Violations does not learn a good solution and has issues learning throughout the experimentation process, this is reflected in TLO Violations - Emissions, which also has difficulty learning. TLO Violations - Cost learns an optimal solution over 20,000 episodes although the learning rate is slow.

In Figure 2, TLO Cost-Violations performs poorly when compared to the other ordering types, while TLO Violations - Cost and TLO Violations - Emissions learn optimal solutions. The results presented in Figures 1, 2 and 3 confirm the order of objectives can have a significant impact on system performance.

5.3 Comparison of Dynamic and Fixed Thresholding Techniques

To evaluate the performance of fixed thresholding compared to dynamic thresholding, we conducted a number of experiments with varying thresholds and compared the results against dynamic

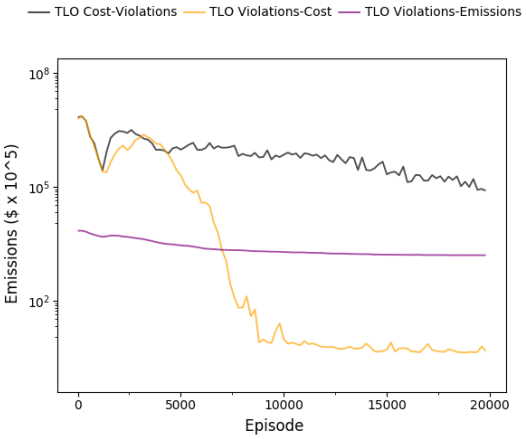

Figure 3: Effect of Ordering of Objectives for Emissions

thresholding. We tested various fixed thresholds in order to convey the level of system knowledge and experimentation that is required to identify and select a useful threshold value. We executed six separate experiments with six different thresholds and contrasted these with other experiment results that used dynamic thresholding. In this experimentation set we optimised objectives in the following order: violations, cost and emissions. This is the best ordering of objectives that was identified in Section 5.2.

Table 1: Fixed Threshold Parameters

Experiment Name	Violations	Cost
TLO-1	0	0
TLO-2	-1×10^8	-2.8×10^8
TLO-3	-3×10^7	-6×10^7
TLO-4	-2×10^7	-3×10^7
TLO-5	-1×10^7	-2.8×10^6
TLO-6	-5×10^6	-2.6×10^6

The fixed thresholds used in the experimentation are specified in Table 1. TLO-1 to TLO-3 are large values that the system produces before learning optimal solutions and, therefore, should have a negative effect on system performance. TLO-4 to TLO-6 are predefined optimal outcomes, with TLO-6 being the most optimal solution. The dynamic threshold experimental results are represented in Figures 4, 5 and 6 as TLO-Global. TLO-Global uses Enhancement 2 that was established in Section 3.3. Enhancement 2 achieved better results in initial experimentation when compared to Enhancement 1 and Enhancement 3. This experimentation set marks the first time fixed and dynamic thresholding techniques have been evaluated together.

Figures 4, 5 and 6 present the effect of each fixed threshold on cost, emissions and violations. From the graphs presented, there is no significant improvement in performance for specifying any particular thresholds.

From Figures 4, 5 and 6 we can deduce that threshold selection is not a determining factor in the final system performance when using TLQ-learning in a multi-objective multi-agent problem domain.

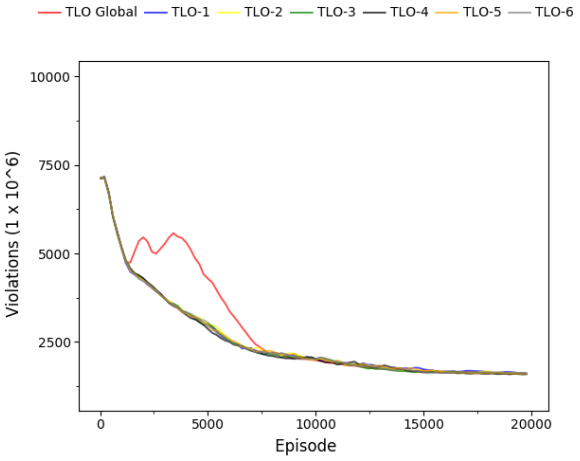

Figure 4: Effect of Fixed Thresholding for Violations

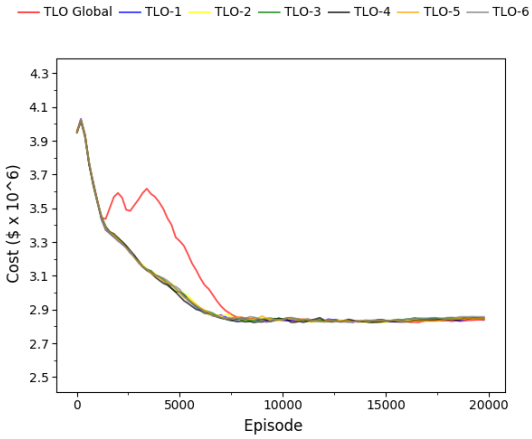

Figure 5: Effect of Fixed Thresholding for Cost

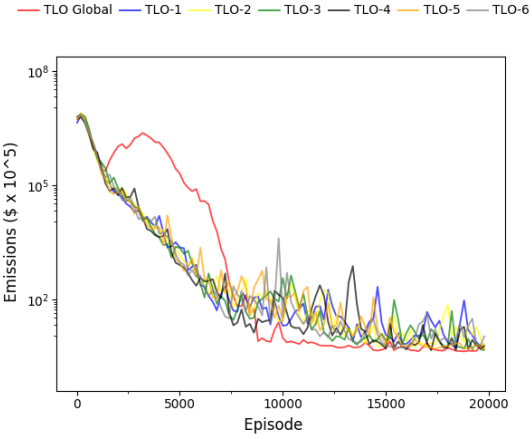

Figure 6: Effect of Fixed Thresholding for Emissions

To determine an optimal solution, specialist knowledge is required with significant trial and error. In this case, it is recommended to use dynamic thresholding with the CIT algorithm. Although performance is not better when compared to fixed thresholding techniques, CIT removes the need for specialist knowledge.

Using the CIT algorithm can remove the trial and error associated with selecting thresholds. In Figure 6, the TLO-Global (CIT algorithm with TLQ-learning) has better performance than all fixed threshold experiments. In Figures 4 and 5, TLO-Global (CIT algorithm with TLQ-learning) and all fixed threshold experiments perform as well as each other.

5.4 TLQ-learning with CIT vs. linear scalarisation

In this section we compare standard benchmark reinforcement learning scalarised rewards $G(+)$ and $D(+)$ with our CIT algorithm with TLQ-learning (TLO-Global). Note that $D(+)$ is included here only to show best case performance in this domain when perfect global information is available to all agents during learning; $D(+)$ uses far more information than our TLO-Global approach as it calculates an individually shaped reward signal for each agent in the system, whereas TLO-Global does not. In this section, we are interested in whether our dynamic TLO approach can achieve better performance than the scalarised global reward with hand tuned weights. Comparing our TLO-Global with the $D(+)$ reward is however crucial to understanding the performance compared to the state-of-the-art results, but it is not a fair comparison. All plots for cost, emissions and violations are based on a 200 point average across 50 statistical runs. All claims of statistical significance are supported by two-tailed t-tests assuming unequal variances, with $p = 0.05$ selected as the threshold for significance.

Table 2: DEED Average Solutions

	Cost ($ $\times 10^6$)	Emissions (lb $\times 10^5$)	Violations
$G(+)$	2.8712	4.4183	1594.2466
TLO - Global	2.8305	4.2107	1612.6165
$D(+)$	2.7079	3.9822	1624.4449
NSGA-II [2]	2.5226	3.0994	-
PSO-AWL [9]	2.5463	2.9455	-

In Figures 7, 8 and 9, the $D(+)$ reward performs as expected converging to a stable policy in less than 500 episodes. The $G(+)$ learns a good policy, it takes the $G(+)$ reward over 5,000 episodes to converge.

The results generated in this experimentation set replicate the findings of Mannion et al. [8], where the $D(+)$ reward is statistically better than $G(+)$. The aim of this section is to evaluate the TLO-Global reward against the linear scalarised rewards for cost, emissions and violations. The linear scalarised reward structures perform as expected across all objectives. TLO-Global also reaches a lower cost than the $G(+)$ reward and converges to a similar violations result as $D(+)$ and $G(+)$. Finally, TLO-Global reaches a lower emissions value when compared to $G(+)$.

There is no statistical significance when comparing the mean performance of TLO-Global and $D(+)$ for the cost objective. The

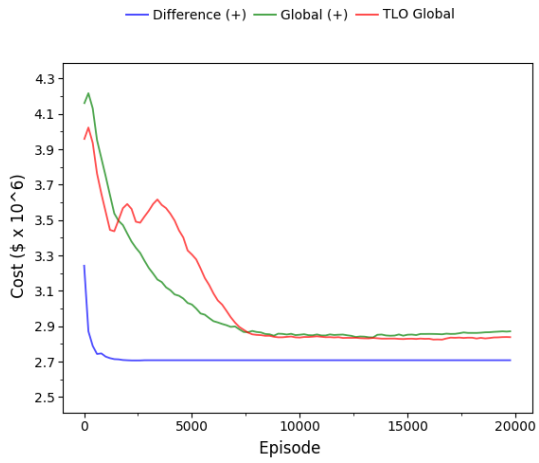

Figure 7: Cost with Linear Scalarisation

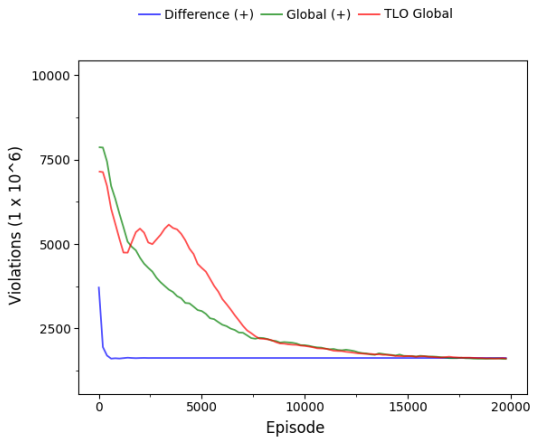

Figure 9: Violations with Linear Scalarisation

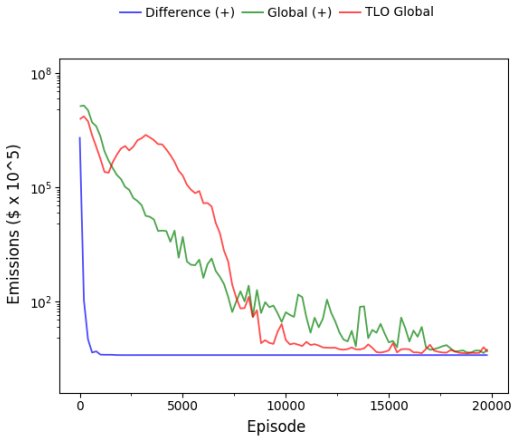

Figure 8: Emissions with Linear Scalarisation

differences in the means between TLO-Global and G (+) are statistically significant for the cost objective ($p = 1.2783 \times 10^{-4}$). There is no statistical difference between TLO-Global and D (+) or TLO-Global and G (+) for the violations objective. For the emissions objective the difference in final means of TLO-Global and D (+) are not statistically significant, while the difference in final means for TLO-Global and G (+) were found to be statistically significant ($p = 0.002895$). Therefore, it can be concluded that our dynamic TLO approach outperforms the G (+) with hand tuned weights from prior work [8] reward on the cost and emissions objectives.

6 CONCLUSION & FUTURE WORK

Multi-objective optimisation appears in various disciplines and is a fundamental mathematical problem. Using linear scalarisation techniques with MARL to find solutions to multi-objective problems has been successful but the limitations are significant. In this paper we have successfully extended TLO to be used with Q-learning in a large MOMARL problem domain. The results we have presented highlight the fact that TLO, used with our dynamic thresholding technique, can exceed the performance of a hand-tuned linear scalarisation approach when learning with the global reward. Using TLO with Q-learning for MOMARL has many possible future applications, such as natural resource management [3] and traffic signal control [5].

With respect to future studies, it would be important to evaluate TLO using thresholds set at a local level by giving each agent its own threshold. This complex experimentation has not been carried out before and it could have benefits to overall system performance. It would also be important to evaluate TLO in a non-cooperative MOMARL environment, as the DEED domain is a cooperative environment. It would also be promising to create a TLO-Difference reward that captures each agents' contribution to the system performance. Given the results for TLO-Global perform better than the $G(+)$ reward it could be possible to create a reward structure that performs better or at least as well as the $D(+)$ reward.

Although our results are promising there are some limitations; the objectives of violations, cost and emissions in the DEED problem domain are correlated. Optimising one objective results in optimising the other objectives, while the exact degree of correlation is unknown, which was noted during experimentation. We expect even more significant results could be demonstrated in other MOMARL domains where the objectives have a lower degree of correlation. We also plan to include more thorough results of experimentation for the CIT enhancements in a later publication as these results are only briefly mentioned in this paper.
Finally, setting thresholds with TLQ-learning is a more intuitive process when compared to setting weights using linear scalarisation. Using our dynamic thresholding technique can also remove the requirement for user expertise completely when using TLO in MOMARL domains.

ACKNOWLEDGMENTS

Conor F. Hayes is funded by the National University of Ireland Galway Hardiman Scholarship.

REFERENCES

[1] M. Babbar-Sebens and S. Mukhopadhyay. 2009. Reinforcement learning for human-machine collaborative optimization: Application in ground water monitoring. In *2009 IEEE International Conference on Systems, Man and Cybernetics*. 3563–3568. https://doi.org/10.1109/ICSMC.2009.5346708

[2] M Basu. 2008. Dynamic economic emission dispatch using nondominated sorting genetic algorithm-II. *International Journal of Electrical Power and Energy Systems* 78 (02 2008), 140–149. https://doi.org/10.1016/j.ijepes.2007.06.009

[3] C Bone and S Dragićević. 2009. GIS and Intelligent Agents for Multiobjective Natural Resource Allocation: A Reinforcement Learning Approach. *Transactions in GIS*. 13, 3 (2009), 253–272.

[4] Z. Gabor, Z. Kalmar, and C. SzepesvariA. 1998. Multi-criteria reinforcement learning. *In The fifteenth international conference on machine learning, pp. 197–205* (1998).

[5] D Houli, L Zhiheng, and Z Yi. 2010. Multiobjective Reinforcement Learning for Traffic Signal Control Using Vehicular Ad Hoc Network. *EURASIP Journal on Advances in Signal Processing* 2010, 1 (16 Sep 2010), 724035. https://doi.org/10.1155/2010/724035

[6] Ishfaq A, Sanjay R, and Samee Ullah Khan. 2008. Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. 1–6. https://doi.org/10.1109/IPDPS.2008.4536420

[7] C. Li and K Czarnecki. 2019. Urban Driving with Multi-Objective Deep Reinforcement Learning. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2019).

[8] P Mannion, K Mason, S Devlin, E Howley, and J Duggan. 2016. Multi-Objective Dynamic Dispatch Optimisation using Multi-Agent Reinforcement Learning. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2016).

[9] K Mason, J Duggan, and E Howley. 2017. Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants. *Neurocomputing* 270 (2017), 188 – 197. https://doi.org/10.1016/j.neucom.2017.03.086 Distributed Control and Optimization with Resource-Constrained Networked Systems.

[10] A.E Smith, D.W Coit, T Baeck, D Fogel, and Z Michalewicz. 2000. Penalty functions. (2000).

[11] P. Vamplew, R Dazeley, A. Berry, R. Issabekov, and E. Dekker. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning, 84(1-2), pp.51-80* (2011).

[12] K Van Moffaert, M.M Drugan, and A Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. (2013).

[13] C Watkins and P Dayan. 1992. Q-Learning. (1992).

[14] M Wiering and M van Otterlo. 2012. Reinforcement Learning and Markov Decision Processes. (2012).

[15] David H Wolpert, Kevin R Wheeler, and Kagan Tumer. 2000. Collective intelligence for control of distributed dynamical systems. *EPL (Europhysics Letters)* 49, 6 (2000), 708.

[16] L Yliniemi and K Tumer. 2014. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Simulated Evolution and Learning*. Springer International Publishing, 407–418.

[17] L.M. Zintgraf, Roijers, Linders D.M., C.M. S., Jonker, and A Nowé. 2016. Ordered preference elicitation strategies for supporting multi-objective decision making. *17th International Conference on Autonomous Agents and Multi-Agent Systems, pp.1477–1485* (2016).