Analyzing the Robustness of Multi-Objective Reinforcement Learning Algorithm to Dynamics and Hyperparameters

Tai Dang
Manning College of Information & Computer Sciences
University of Massachusetts, Amherst
ttdang@umass.edu

Abstract

In this paper, we propose a method for analyzing the effect of changing hyperparameters on the set CCS(Π) in the context of multi-objective reinforcement learning. Specifically, we investigate how the policies produced by the Optimistic Linear Support (OLS) algorithm are affected by changes in the objective weights or other parameters of the MDP, such as the gravity or object position. We aim to answer questions such as: How do different values of gamma influence different properties of the results? Does the CCS set expand or shrink as we change gamma? Does the frequency of policy changes increase as we change gamma? In what cases do most policies tend to occupy similar regions of the space of objective weights?

1 Introduction

In many real-world situations, we must consider multiple factors when making decisions. For example, when managing a water reservoir with a hydroelectric power plant, we may want to maximize energy production, minimize irrigation shortages, and minimize the risk of flooding (Reddy and Kumar, 2006; Pianosi et al., 2013; Castelletti et al., 2013). Likewise, in the medical field, we may want to maximize the effectiveness of treatment while minimizing potential side effects (Jalalimanesh et al., 2017; Lizotte et al., 2010). Essentially, most decision-making situations in the real world involve multiple objectives.

Most decision-making algorithms are designed to optimize a single objective (Sutton and Barto, 2005), even though real-world decision problems often involve multiple objectives. One approach to addressing this issue is to combine all the relevant objectives into a single scalar reward function, such as the Optimistic Linear Support (OLS) algorithm (Roijers, 2016). However, the accuracy of this approach can vary depending on the specific application, with some situations requiring more precise policies (e.g., medical treatment or water management) and others allowing for more flexible, non-deterministic policies (e.g., chess). This article introduces a method for analyzing the set CCS(Π) produced by a multi-objective decision-making algorithm, focusing on comparing and adapting policies for different applications. To the best of our knowledge, this is the first time this topic has been explored in detail. We hope this article will serve as a foundation for future research in policy analysis.

2 Methodologies

In this section, we will introduce the basic multi-objective sequential decision problem, the OLS algorithm, and our method of analysis policy.

2.1 Multi-Objective Problem setting

We use a multi-objective Markov decision process (MOMDP) (Hayes et al., 2021) to formalize a decision-making problem involving multiple objectives. While more complex models, such as multi-objective partially observable Markov decision processes and multi-objective multi-agent systems, also exist (Soh and Demiris, 2011; Nian et al., 2020), the MOMDP is a practical and relatively straightforward framework for studying multi-objective decision-making. This article focuses on single-agent MOMDPs and does not consider more complex models. A MOMDP is represented by the tuple $\langle S, A, T, \gamma, \mu, \mathbf{R} \rangle$ where:

- S: is the state space
- A: is the action space
- T: S × A × S → [0, 1] is a probabilistic transition function
- $\gamma \in [0, 1)$ is a discount factor
- μ: S → [0, 1] is a probability distribution over initial states

- **R**: S × A × S → R^d is a vector-valued reward function specifying the immediate reward for each of the considered d ≥ 2 objectives

The main difference between a single-objective Markov Decision Process (MDP) and a Multi-Objective Markov Decision Process (MOMDP) is the reward function **R**, which is a vector-valued function that provides a numeric feedback signal for each objective being considered. The length of this reward vector is equal to the number of objectives in the MOMDP.

2.2 Optimistic Linear Support Algorithm

When an agent's preferences over objectives can be represented linearly, an optimal solution corresponds to a convex coverage set (CCS). This means that, for any new set of linear preferences, there exists at least one optimal policy within the CCS. A popular method used for producing the CCS set is the OLS algorithm. The Optimistic Linear Support (OLS) algorithm [1] is a state-of-the-art method for constructing a CCS through the iterative learning of policies that optimize different linear preferences (Roijers et al., 2015; Mossalam et al., 2016).

OLS approaches multiple objectives in the outer loop by utilizing a single-objective method as a subroutine and constructing the CCS incrementally. During each iteration of the outer loop, OLS may add at most one new vector to the partial CCS, denoted as S. To identify this vector, OLS selects a linear scalarization weight vector **w** that offers the maximum potential improvement, which is defined as an upper bound on the difference between the scalarized value function, $V_S^*(\mathbf{w})$, and the optimal scalarized value function, ${V_{\text{CCS}}}^*(\mathbf{w})$. Once OLS has determined **w**, it passes it to the inner loop. In the inner loop, OLS invokes a problem-specific single-objective solver to solve the scalarized single-objective decision problem that results from applying **w** to the MODP. The optimal policy for this scalarized problem and its corresponding value vector is then added to the partial CCS.

OLS operates through a series of iterations in which it selects a **w** and utilizes this weight to determine the maximal possible improvement at each iteration. After a finite number of iterations, OLS can terminate as no further improvements can be made to the partial CCS. This allows OLS to function as a generic multi-objective method.

[1] Refer to (Roijers et al., 2015) sec. 3.3 for implementation of the OLS

OLS takes advantage of the optimal value function property of $V_S^*(\mathbf{w})$. OLS gradually constructs the CCS by adding new value vectors discovered at special weights called corner weights S to an initially empty partial CCS. These corner weights are weights at which VS(w) changes slope in all directions and, therefore, must be weights at which $V_S^*(\mathbf{w})$ and the optimal policy set, $\Pi_S(\mathbf{w})$, consist of multiple value vectors and corresponding policies. Each corner weight is prioritized based on the maximum potential improvement of discovering a new value vector at that corner weight. Once the maximum potential improvement becomes zero, OLS recognizes that the partial CCS is complete.

2.3 Policy Analysis Method

Given the set of CCS produced by the OLS algorithm in section 2.2, for each weight, we will pick the best-fit policy in the CCS set for analysis purposes. This can be done by changing the objective weight. We denote the variable $StepSize$ = 0.001, which is the smallest step in which we will increase or decrease the objective weight when conducting analysis. We begin by setting one objective to 1 and all the others to 0. We increment or decrement each objective by $StepSize$ and repeat the process until we have tried all possible combinations. For each weight, we choose the policy with the highest scalarized reward, which is calculated by taking the dot product of the weight and the policy's discounted vectors in the CCS. We then plot that policy based on the given weight, using the objectives as axes on the graph. Since the last objective weight can depend on the other weights, we only use a (d-1)-dimensional graph, where d is the number of objectives in the MOMDP.

We plot the results and observe some interesting behavior in the CCS set. For instance, more policies are present when the γ value is high, and some of these policies occupy minimal areas on the graph. This suggests that sensitive policies require a lot of time and effort to adjust the hyperparameters; therefore, they may be located in small, hard-to-graph areas. We will delve deeper into this analysis in section 3.

We only limit our approach to graphing a maximum of three objective problems because it becomes too expensive to graph more than that. While it is theoretically possible to graph up to four objective problems in three-dimensional space because that is how humans process information,

		R1	E2	
		E1		R2
		H		

Figure 1: A resource-collection domain

			E2	
				R2
	E1			
R1		H		

Figure 2: Modified resource-collection domain

we stay within three objectives due to the difficulties mentioned above. Nevertheless, this remains a significant limitation of our method, and further research will be required to address it.

3 CCS Analysis

In this section, we conduct an analysis of three different environments (continuous and discrete). This analysis is mostly heuristical rather than a proof of concept.

3.1 Two-Objective Problem

3.1.1 Deep Sea Treasure (DST)

In a classic multi-objective reinforcement learning (MORL) benchmark (Yang et al., 2019), an agent is tasked with navigating a 10 x 11 grid world while balancing the trade-off between time cost and treasure value. The grid world contains 10 treasures of varying values, with the value increasing as the distance from the starting point (0, 0) increases. It has been established that the Pareto frontier of this environment is convex.

3.1.2 DST Analysis

Examination of Fig. 3 reveals that the value of γ significantly impacts the CCS set. This is particularly evident in the range [0.9,1), where most policies are concentrated. A clear pattern emerges whereby higher values of γ correspond to a greater number of policies, many of which are specialized in solving very narrow regions of the space of $\mathbf{w}$. High values of γ can be utilized to identify the best policies in extreme areas. As the value of γ decreases, the CCS set also decreases in size. Still, the area occupied by each policy becomes more evenly distributed, with the number of policies remaining at two to three. Based on these observations, it can be inferred that values of γ in the range [0.4,0.9] result in policies that occupy similar areas in the space of $\mathbf{w}$. As γ approaches 0, one policy dominates the others and eventually becomes the sole member of the CCS set. While we do not have a concrete explanation for this behavior, we speculate that it may be related to training time, as higher values of γ can lead to longer training times and, therefore, a higher likelihood of discovering new sets of policies.

3.2 Three-Objective Problem

3.2.1 Resource Gathering Environment

In this environment, (Barrett and Narayanan, 2008), an agent is placed in a grid 1 and must navigate around to gather resources. The agent starts at the home base, labeled H, and can move in the four cardinal directions. The goal is for the agent to collect resources from locations R1 and R2, and bring them back to the home base, H. If the agent can successfully return to the home base with one or both of the resources, it will receive a reward. However, if the agent steps on one of the enemy spaces labeled E1 or E2, there is a chance that it will be a 10% probability of being attacked, which will result in a penalty and the loss of any resources the agent was carrying. The agent's reward space consists of the potential outcomes of being attacked or bringing back one or both resources. Its reward space is then [enemy, resource1, resource2], so it can get a penalty of [-1,0,0] for being attacked or a reward of [0,1,0], [0,0,1], or [0,1,1] for bringing back one or both resources.

For this problem, we also consider another environment where we modify the position of R1 and E1 to analyze changes in the CCS set when tuning parameters. The modified environment is in Figure

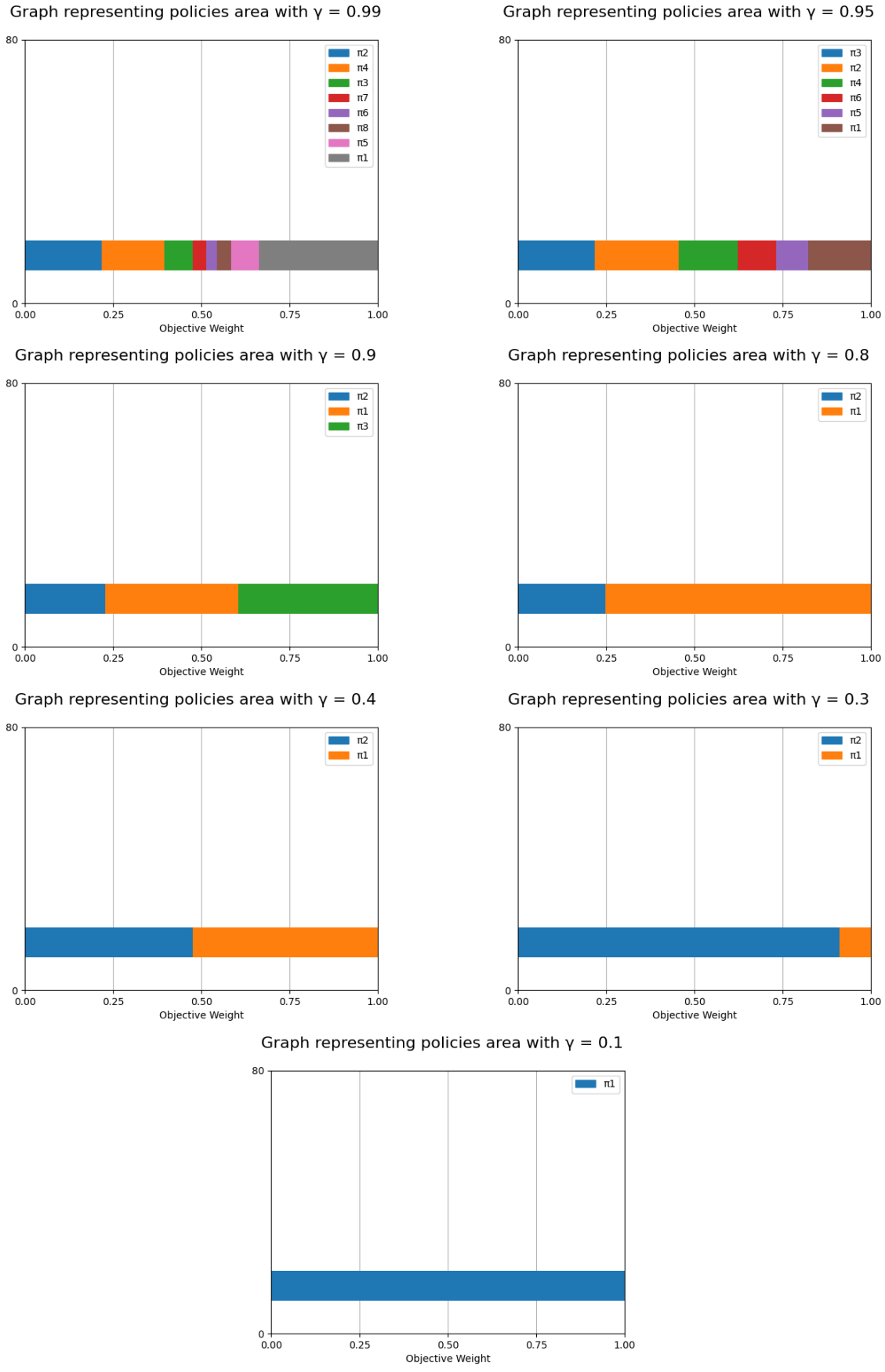

Figure 3: Two-objective Graphs with different γ values for Deep Sea Treasure

2

3.2.2 MO-Mountain Car Environment

In the Mountain-Car task, (Vamplew et al., 2011), a car must escape a valley by reversing up the left side and building enough potential energy to overcome gravity and reach the right side. The inputs for this task are the current position and velocity of the car, and there are three actions the vehicle can take: full throttle forward, full throttle backward, and zero throttle. In the single-objective version, the car receives a penalty of -1 on each step until it reaches the goal.

In the multi-objective version, two additional objectives are added: minimizing the number of times, the car has to reverse and minimizing the number of acceleration actions. A penalty of -1 is received in the corresponding element of the reward vector each time one of these actions is taken. In the single-objective version, the zero-throttle action is rarely beneficial, but in the multi-objective version, the decision of when to use zero-throttle becomes a critical factor in determining the car's policy. This task was modified to a multi-objective case to test the generality of MORL systems, which are designed to handle more than two objectives.

For this problem, gravity is also a critical factor, so we conduct these experiments with three different gravities: 0.0025 (default), 0.025, and 0.00025.

3.2.3 Resource Gathering Analysis

In Fig. 4, which depicts standard resource gathering, there is a pattern similar to that observed in the Deep Sea Treasure environment, with a greater number of policies falling within the range [0.9,1). Interestingly, when the value of γ is close to either 1 or 0, there is a dominant policy that effectively specializes in solving a narrow range of $\mathbf{w}$ values, while the remaining area becomes specialized in solving a more limited range of values. This phenomenon is also observed in the modified environment depicted in Fig. 5, where middle γ values result in policies with more evenly-distributed areas. However, the emergence of a dominant policy occurs earlier in the modified environment compared to the standard environment. For instance, at $\gamma = 0.4$ in the modified environment, there is only a tiny region of blue underneath the green zone, while in the standard setting, the corresponding value is less than 0.2. Further experimentation in additional domains yielded similar patterns, leading us to conclude that altering the environment may affect the algorithm's performance, but it does not significantly alter the behavior of the CCS set.

3.2.4 MO-Mountain Car Analysis

As shown in Fig. 6, a dominant policy(es) is always present. We suspect that this behavior may be influenced by the continuous environment, specifically the x-axis or velocity parameters in the environment. During the graphing process, we observed that policies that are dominated often do not appear on the graph due to the minimal area they occupy, even when the $StepSize$ variable is set to < 0.0001. Furthermore, the highest number of policies typically occurs when γ is relatively high, although this trend is not always consistent, as there are more policies at $\gamma = 0.7$ than at $\gamma = 0.9$. It is also worth noting that when running this experiment multiple times, the CCS set may vary in terms of the number of policies it contains, but the overall trend remains unchanged.

A significant change in the CCS set is evident in Fig. 7, which represents the case where the gravity is reduced by a factor of ten. Similarly to the standard environment, the policies being dominated either do not appear on the graph or occupy a tiny area. One notable difference is the increased number of policies for γ values within the range [0.9,1) or less than 1, compared to the middle γ values. This suggests that reducing gravity significantly impacts the distribution of policies within the CCS set and completely changes the behavior of the CCS set.

In contrast, Fig. 8 shows the opposite trend when the gravity is increased by a factor of ten. Similar to the previous environments, hidden policies are present in the graph for every γ value. However, the CCS set grows the most when γ values are in the mid-range, with seven policies present in the CCS when γ equals 0.6. In contrast, only a single policy is present when γ is close to 1 or 0. This suggests that increasing the gravity has a different effect on the distribution of policies within the CCS set compared to the case where gravity is reduced.

4 Conclusion

We showed that changing hyperparameters of the OLS algorithm or the dynamics of the domain (MDP) itself influences the graphs we generate. Based on the results presented above, it appears that the value of γ has a significant impact on the distribution of policies within the CCS set. In general, higher values of γ are associated with a greater

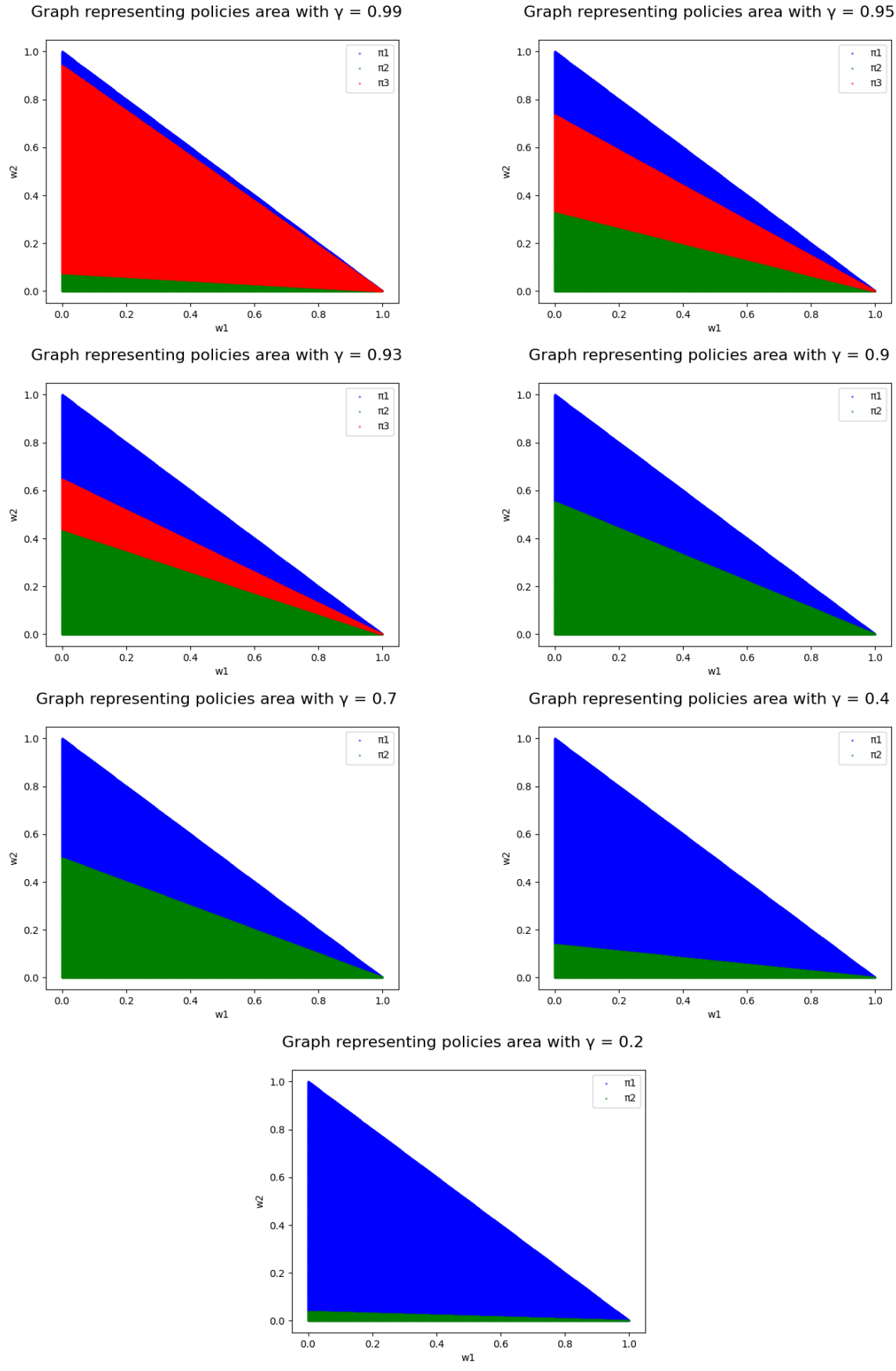

Figure 4: Three-objective Graphs with different γ values for standard Resource Gathering

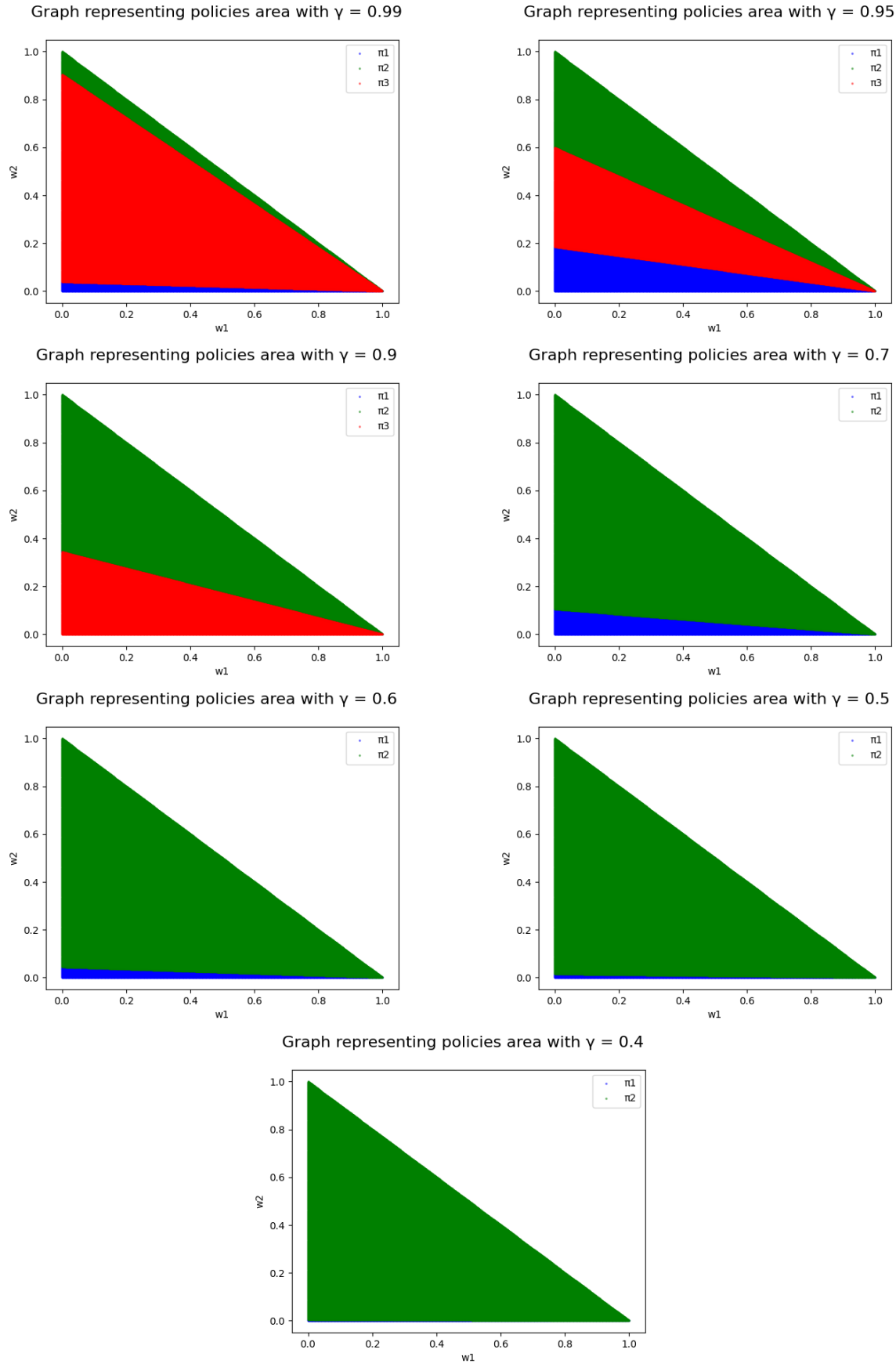

Figure 5: Three-objective Graphs with different γ values for modified Resource Gathering

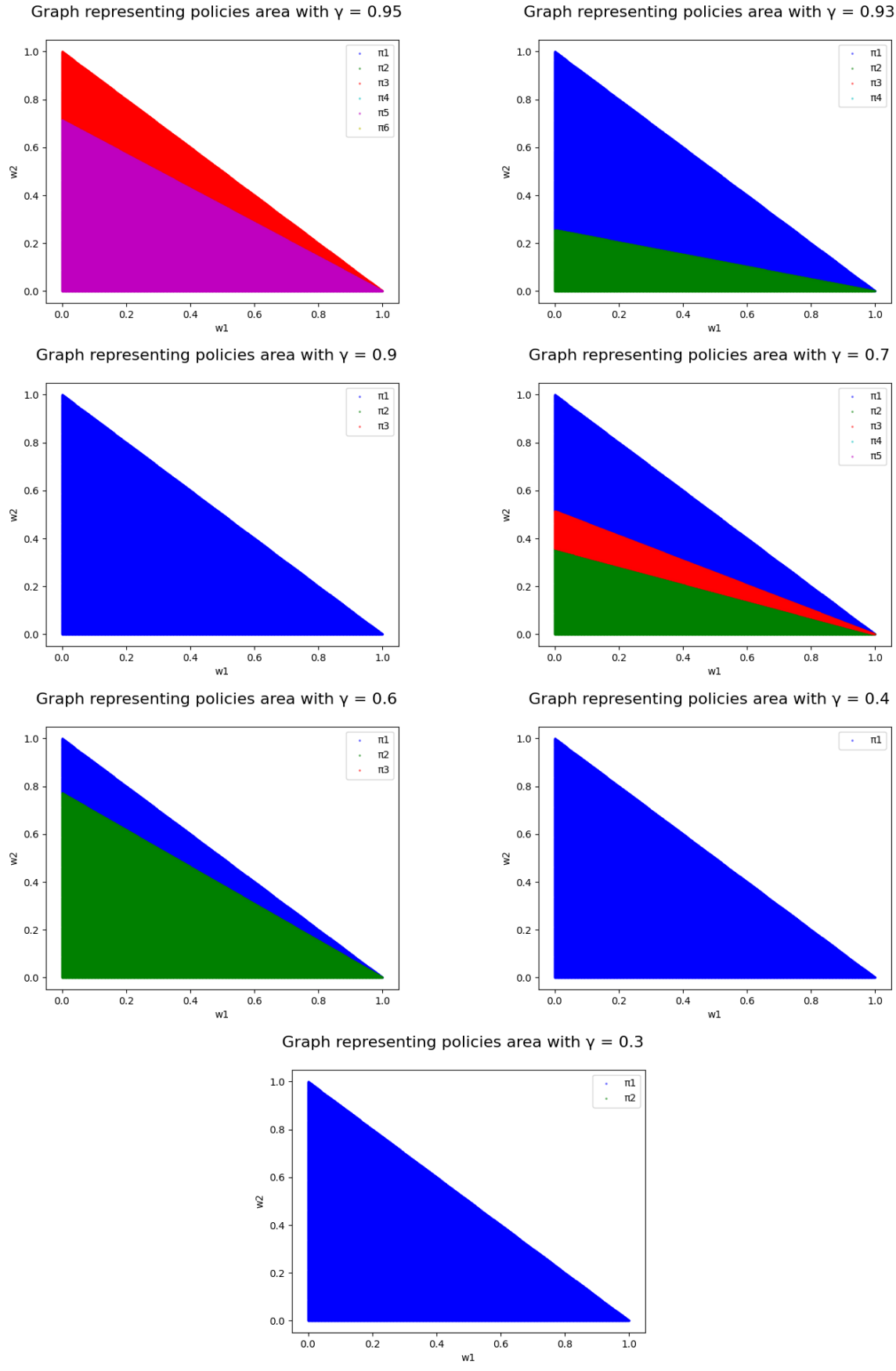

Figure 6: Three-objective Graphs with different γ values for standard MO-Mountain Car

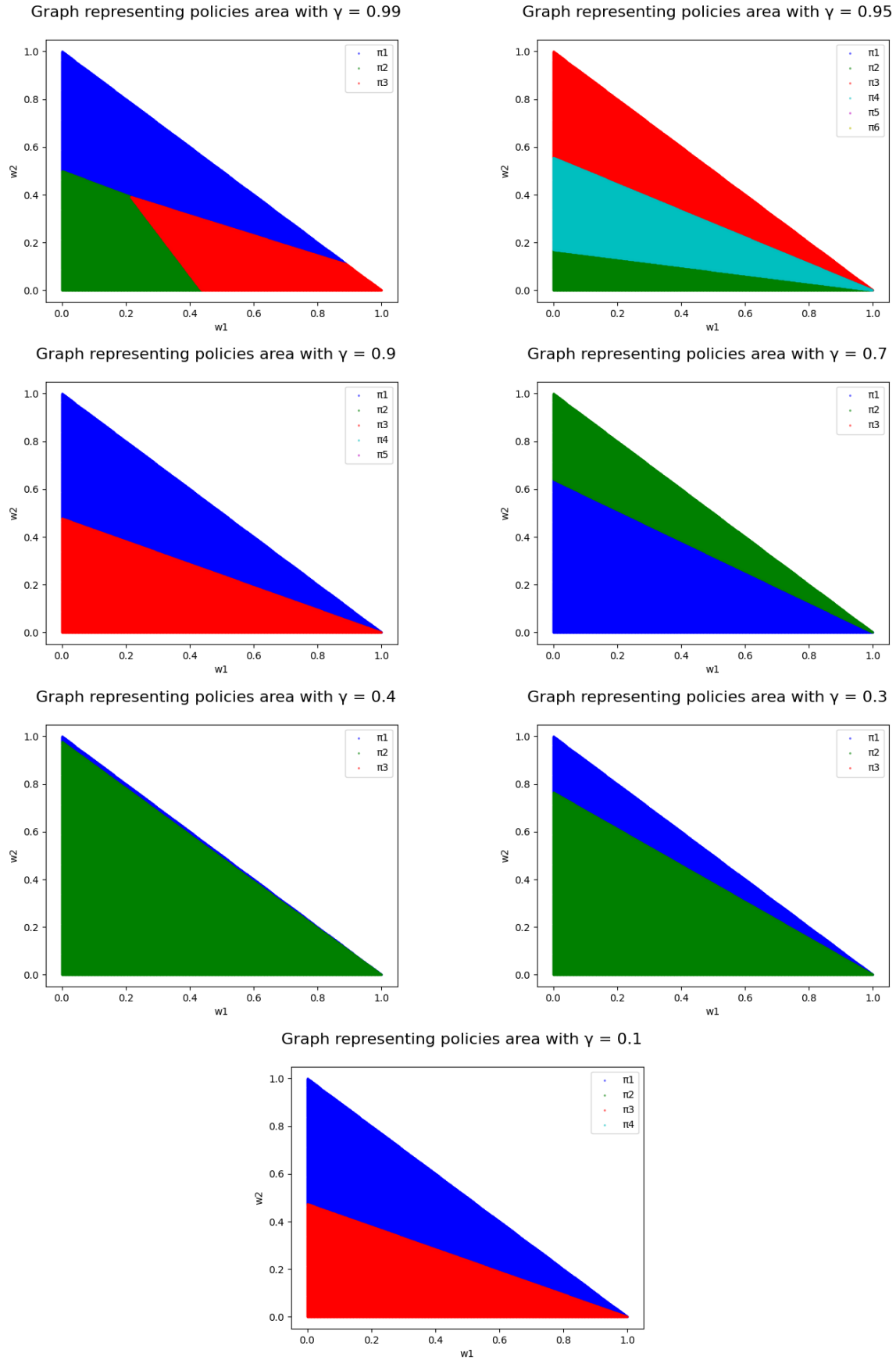

Figure 7: Three-objective Graphs with different γ values for MO-Mountain Car with gravity = 0.00025

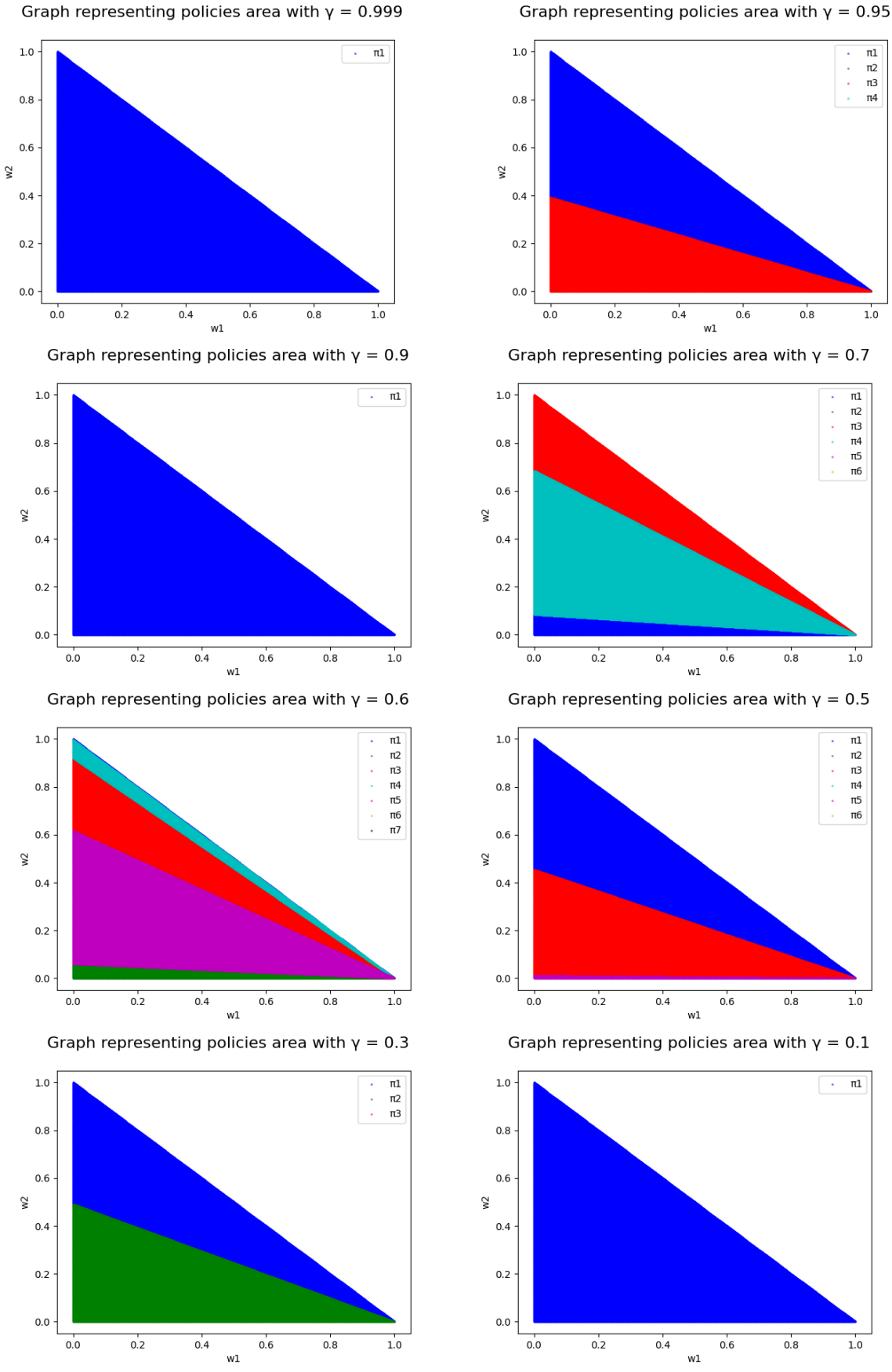

Figure 8: Three-objective Graphs with different γ values for MO-Mountain Car with gravity = 0.025

number of policies, many of which are specialized in solving very narrow regions of the space of $\mathbf{w}$. As the value of γ decreases, the CCS set becomes smaller, but the area occupied by each policy becomes more evenly distributed. Additionally, the presence of a dominant policy is a common feature across all environments studied, although the specific value of γ at which this occurs varies depending on the environment. Modifying the environment, such as by altering the gravity, also appears to have an impact on the distribution of policies within the CCS set. In conclusion, the value of γ and the specific characteristics of the environment both play a role in determining the distribution of policies within the CCS set.

References

Leon Barrett and Srini Narayanan. 2008. Learning all optimal policies with multiple criteria. In *International Conference on Machine Learning*.

Andrea Castelletti, Francesca Pianosi, and Marcello Restelli. 2013. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, 49:3476 – 3486.

Conor F. Hayes, Roxana Ruadulescu, Eugenio Bargiacchi, Johan Kallstrom, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai Aravazhi Irissappane, Patrick Mannion, Ann Now'e, Gabriel de Oliveira Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2021. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36:1–59.

Ammar Jalalimanesh, Hamidreza Shahabi Haghighi, Abbas Ahmadi, Hossein Hejazian, and Madjid Soltani. 2017. Multi-objective optimization of radiotherapy: distributed q-learning and agent-based simulation. *Journal of Experimental & Theoretical Artificial Intelligence*, 29:1071 – 1086.

Daniel J. Lizotte, Michael Bowling, and Susan A. Murphy. 2010. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In *International Conference on Machine Learning*.

Hossam Mossalam, Yannis Assael, Diederik M. Roijers, and Shimon Whiteson. 2016. Multi-objective deep reinforcement learning. *ArXiv*, abs/1610.02707.

Xiao Nian, Athirai Aravazhi Irissappane, and Diederik M. Roijers. 2020. Dcrac: Deep conditioned recurrent actor-critic for multi-objective partially observable environments. In *Adaptive Agents and Multi-Agent Systems*.

Francesca Pianosi, Andrea Castelletti, and Marcello Restelli. 2013. Tree-based fi tted q-iteration for multi-objective markov decision processes in water resource management.

M. Janga Reddy and D. Nagesh Kumar. 2006. Optimal reservoir operation using multi-objective evolutionary algorithm. *Water Resources Management*, 20:861–878.

Diederik M. Roijers. 2016. Multi-objective decision-theoretic planning.

Diederik M. Roijers, Shimon Whiteson, and Frans A. Oliehoek. 2015. Computing convex coverage sets for faster multi-objective coordination. *J. Artif. Intell. Res.*, 52:399–443.

Harold Soh and Y. Demiris. 2011. Evolving policies for multi-reward partially observable markov decision processes (mr-pomdps). In *Annual Conference on Genetic and Evolutionary Computation*.

Richard S. Sutton and Andrew G. Barto. 2005. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286.

Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84:51–80.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Neural Information Processing Systems*.